Big Data

Lecture Notes

Week 14 (2025.06.05)

by

Yuna Jeong

jeongyuna@kisti.re.kr

Final Presentation

- **OFFLINE-only**
  - Location : Kiwoom Hall @ KISTI (245 Daehak-ro, Yuseong-gu, Daejeon)

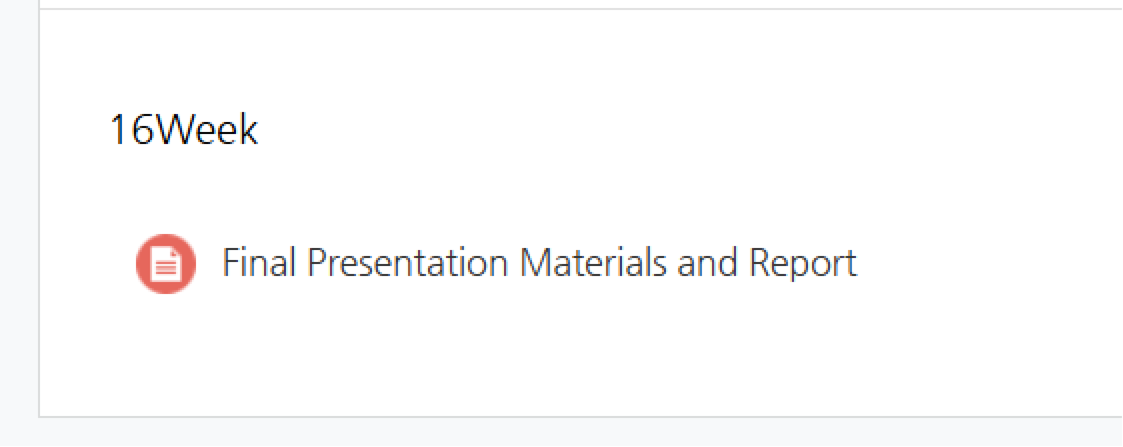
- **Each team has a 15 minute for presentation**
  - Time to present : 10 minutes
  - Question & answer : 5 minutes

  ❖ *If you exceed the time limit, your score will be deducted*

- **Team Order of project presentation:**
  - Team 3 → Team 1

Final Presentation

- **Presentation Material and Report**
  - 'Final Presentation Materials and Report' in Week16 of LMS



16Week

Final Presentation Materials and Report

- **Submission due : 23:55 June 17**
  - Submission punctuality if late, 5 per day
  - Minor edits or supplements to the submitted presentation materials are allowed; however, if changes are made, resubmission is required (no point deduction).

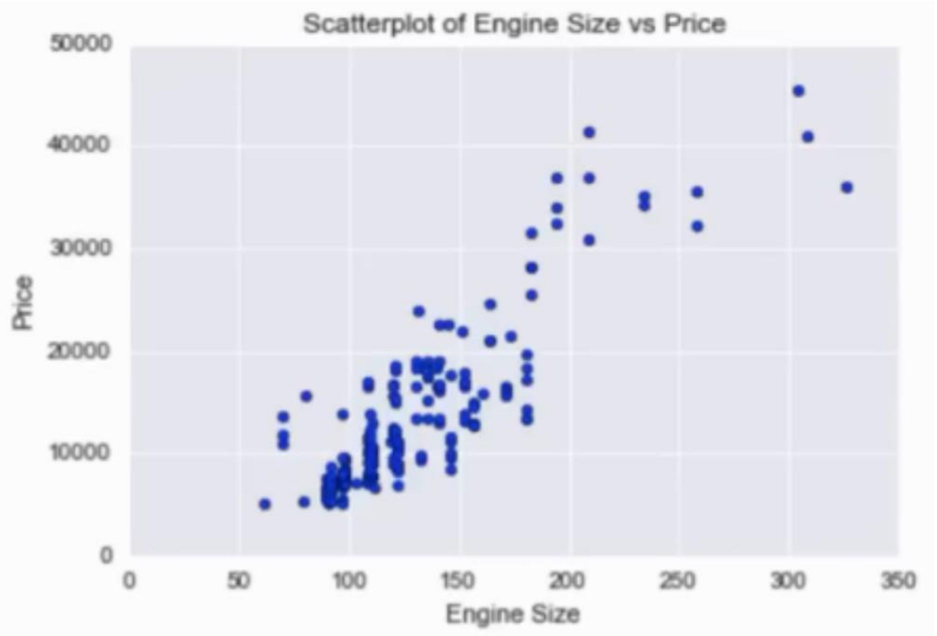
Final Presentation

- **Report**
  - Minimum of five A4 pages
  - Summarizing the entire project, and should include details that were not covered due to the limited presentation time.
  - There is no specific format required, but you may refer to the following items as a guide:
    - Research topic : definition of the problem, discussion on the importance of the problem, definition of the research objectives, …
    - Data science process and results : data collection, understanding, preprocessing, analysis, model selection, training, evaluation, …
    - Insights : insights derived from the results
    - Technical aspects : introduction to the technology stack and tools used
    - Limitations and future plans
    - References : list of papers, documents, or other references used
    - Team member contributions and roles

Data Visualization

- **Data Visualization?**
  - Communicating information through the use of visual elements such as graphs and charts
  - Goal is to make information easy to comprehend, interpret, and retain

- **Two Purpose of Data Visualization**
  - Visualization that effectively delivers your findings to your audience
  - Data analysis : descriptive statistics



Python for Data Visualization

- matplotlib
  - Python 2D plotting library
    - line plots, scatter plots, barcharts, histograms, pie charts etc.
  - Producing publication quality figures in a variety of hardcopy formats
  - A set of functionalities similar to those of MATLAB

  - Relatively low-level; some effort needed to create advanced visualization

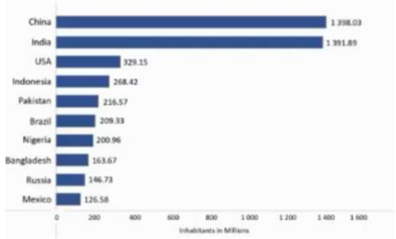
- seaborn
  - Python visualization library based on Matplotlib
  - Provides high level interface for drawing attractive *statistical graphics*
  - Similar (in style) to the popular ggplot2 library in R

Matplotlib vs. Seaborn

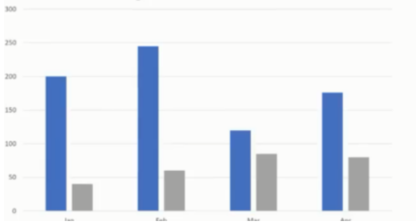
- **Matplotlib** plots various graphs using Pandas and Numpy
- **Seaborn** is the extended version of Matplotlib which uses Matplotlib along with Numpy and Pandas

- **Matplotlib** is highly customizable and powerful
- **Seaborn** avoids a ton of boilerplate by providing default themes which are commonly used

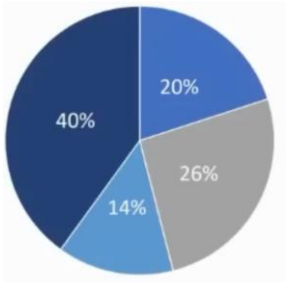
Common Types of Graphs



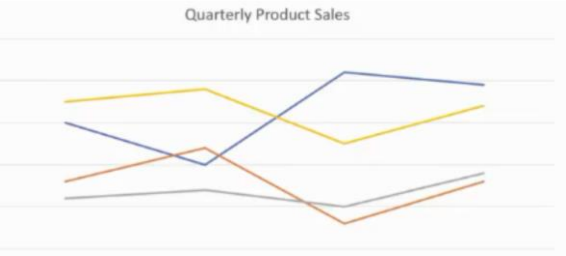
- **Bar Chars**
  - Great for comparing related datasets or parts of a whole



- **Column Charts**
  - Comparing values side-by-side



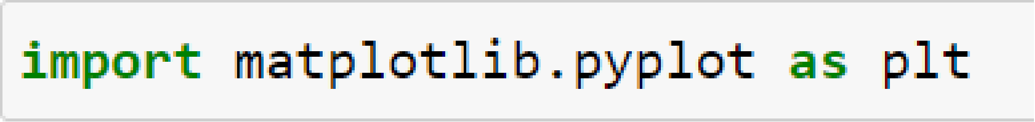
- **Pie Charts**
  - Showing breakdown of an entity into its sub-parts and the proportion of the sub-parts in relation to one another



- **Line Charts**
  - Showing how a data value is changing in relation to a continuous variable

Matplotlib

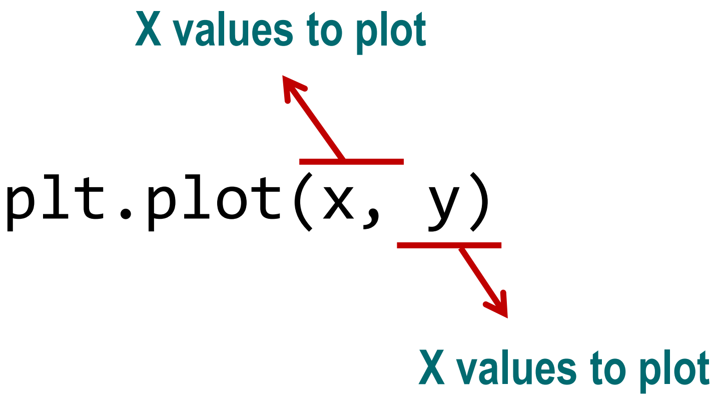
- **matplotlib.pyplot**
  - Collection of functions that make matplotlib work like MATLAB



```python
import matplotlib.pyplot as plt
```

Matplotlib

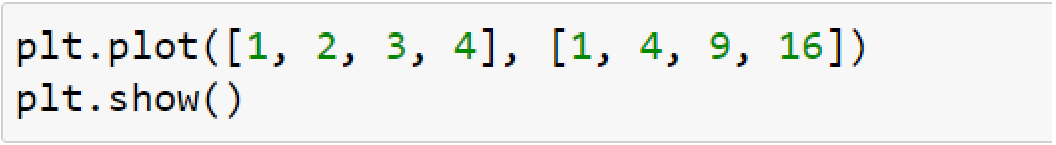
- Line chart : plot()



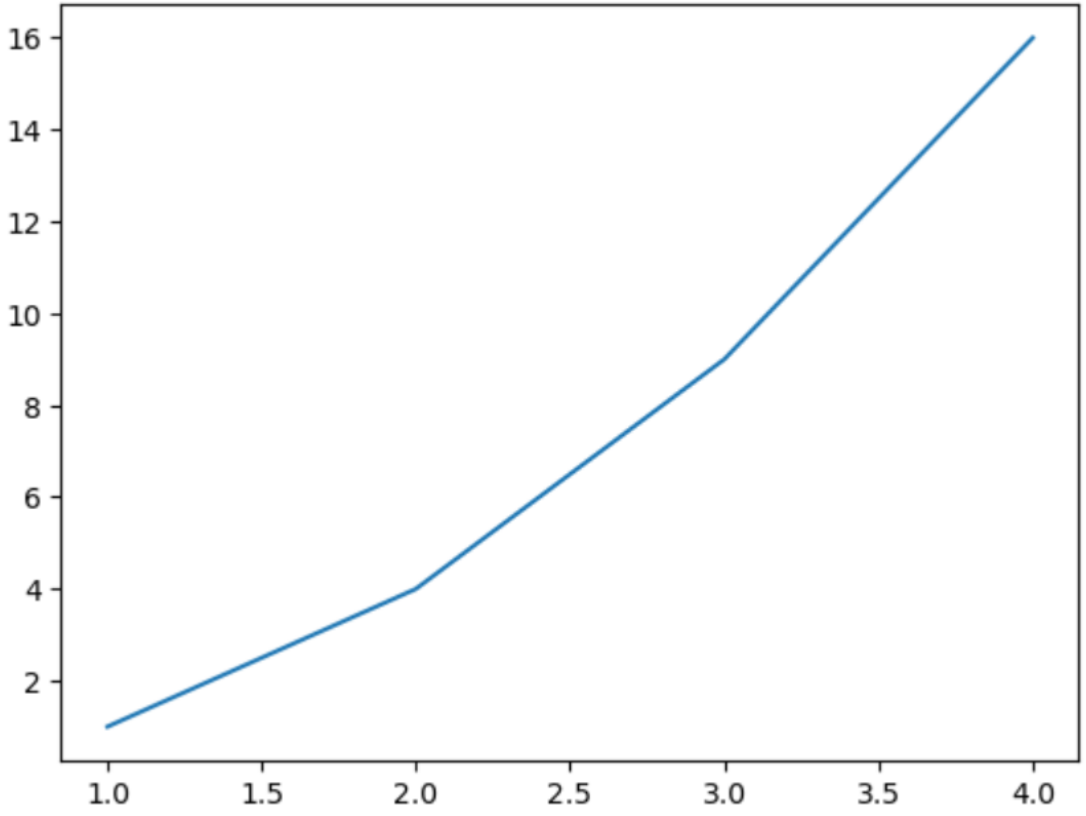
X values to plot

```
plt.plot(x, y)
```

X values to plot

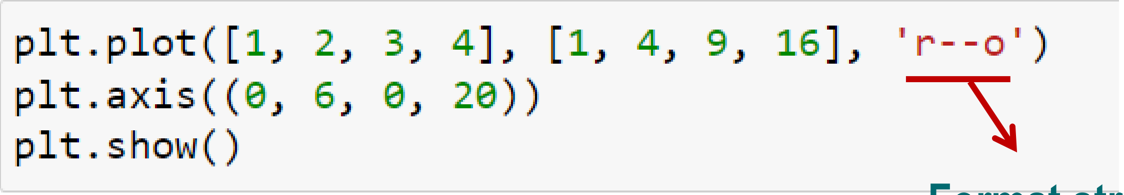


```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
plt.show()
```



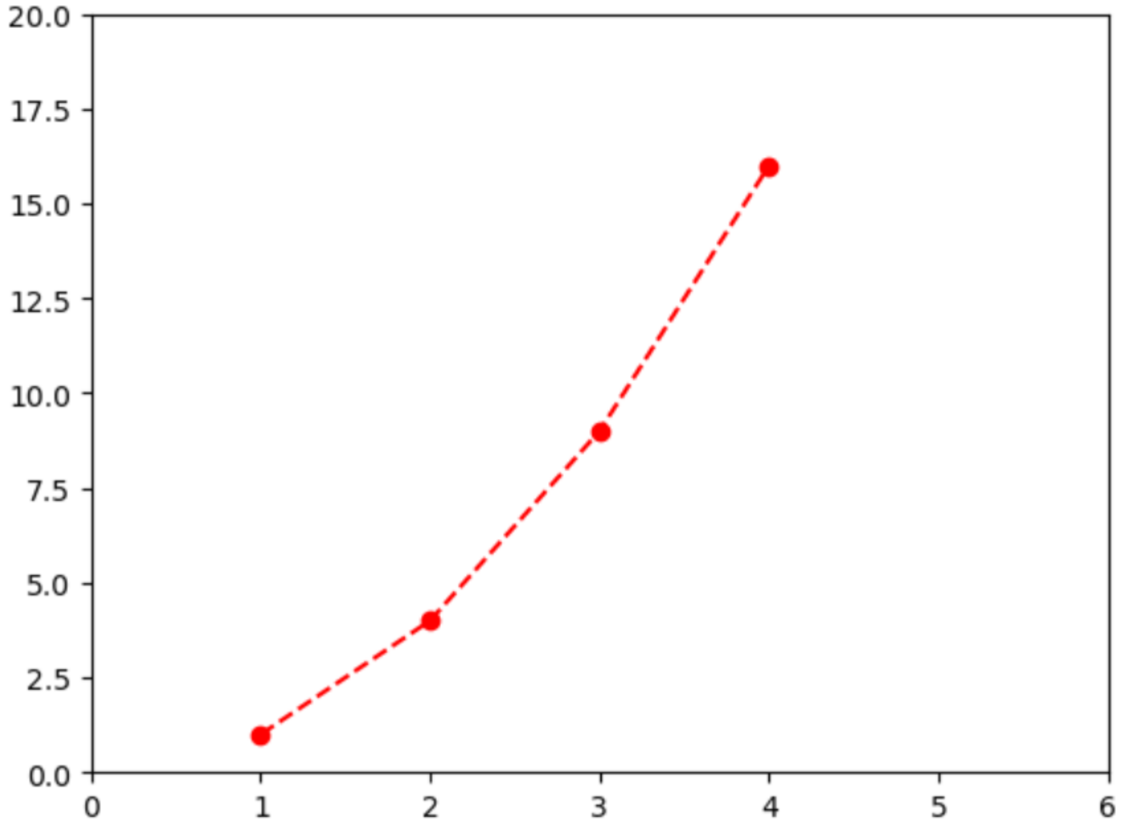
Matplotlib

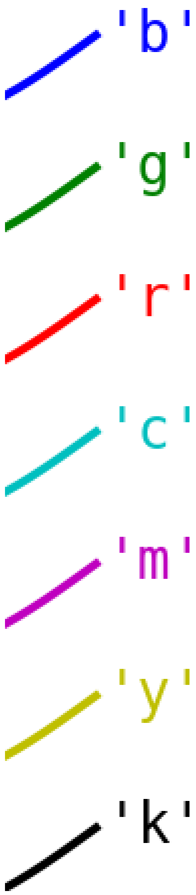
- Line chart : plot()

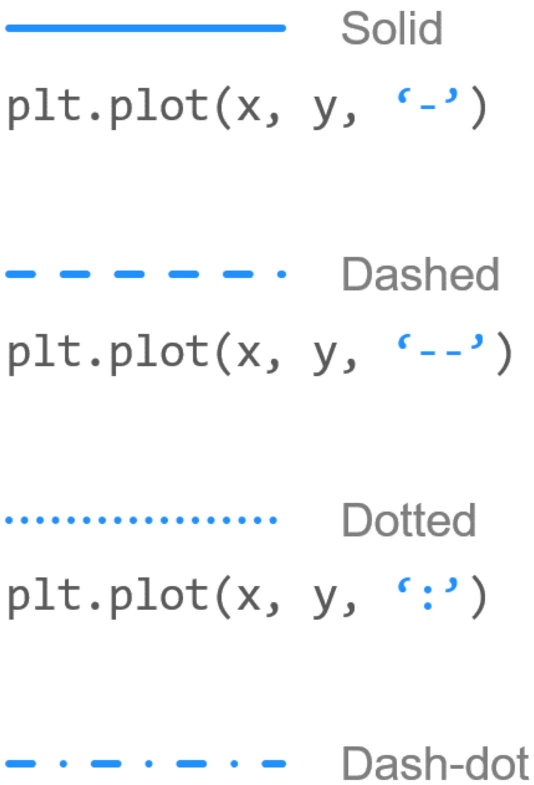


```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'r--o')
plt.axis((0, 6, 0, 20))
plt.show()
```

Format string







plt.plot(x, y, ‘-’)

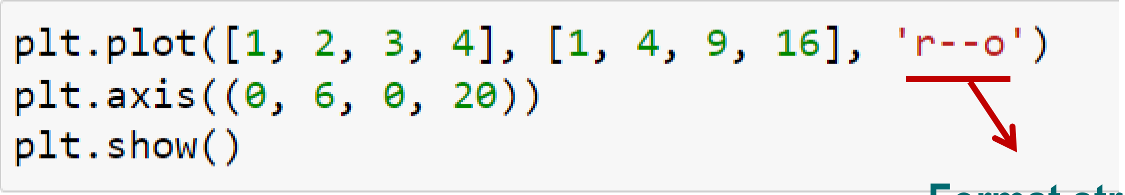
plt.plot(x, y, ‘--’)

plt.plot(x, y, ‘:’)

plt.plot(x, y, ‘-.’)

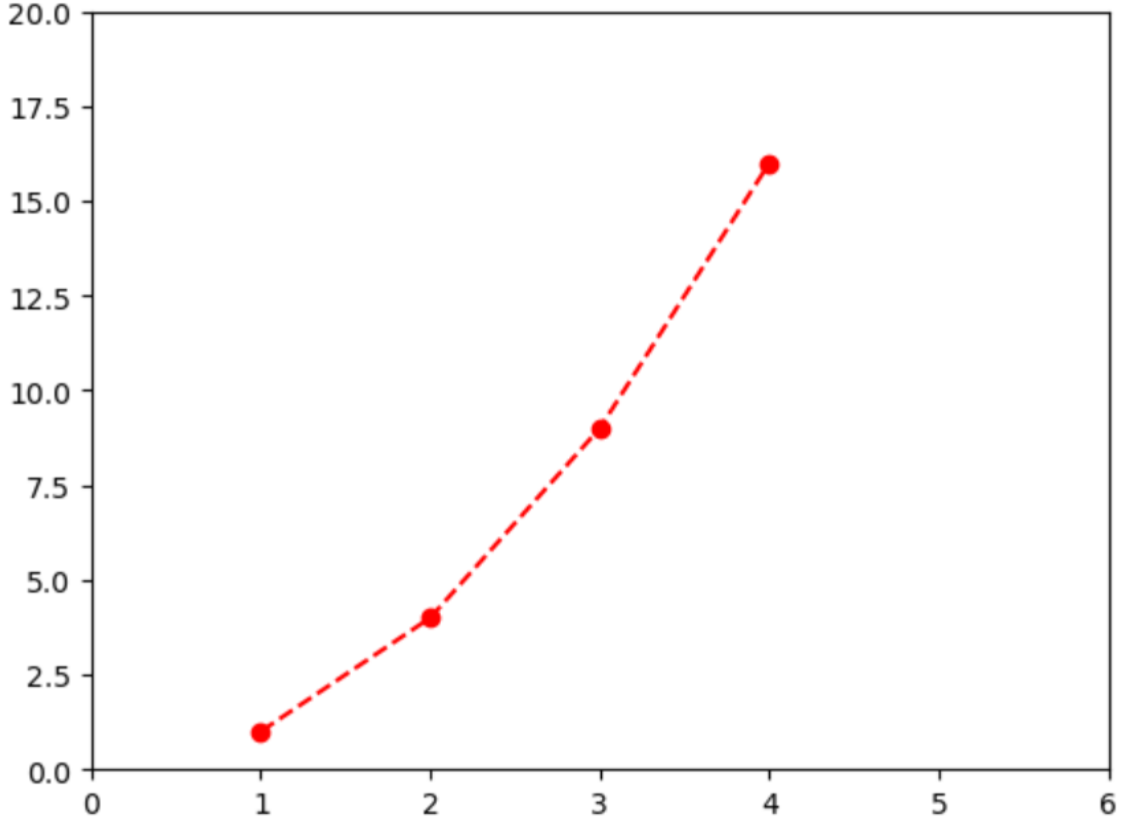
Matplotlib

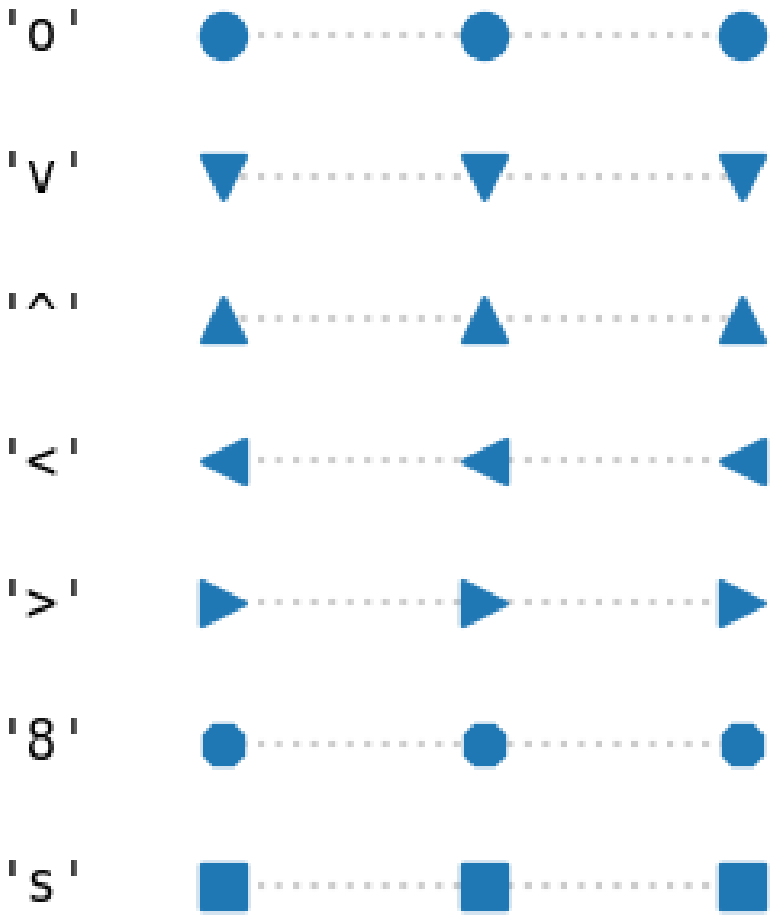
- Line chart : plot()



```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'r--o')
plt.axis((0, 6, 0, 20))
plt.show()
```

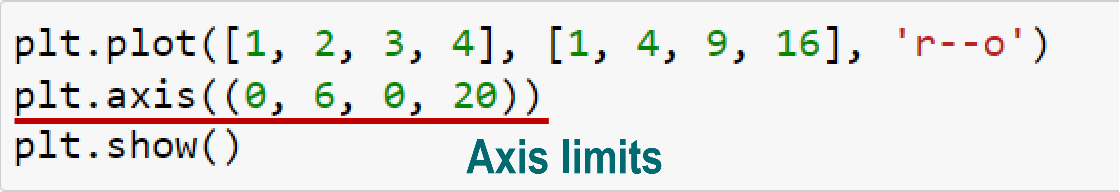
Format string





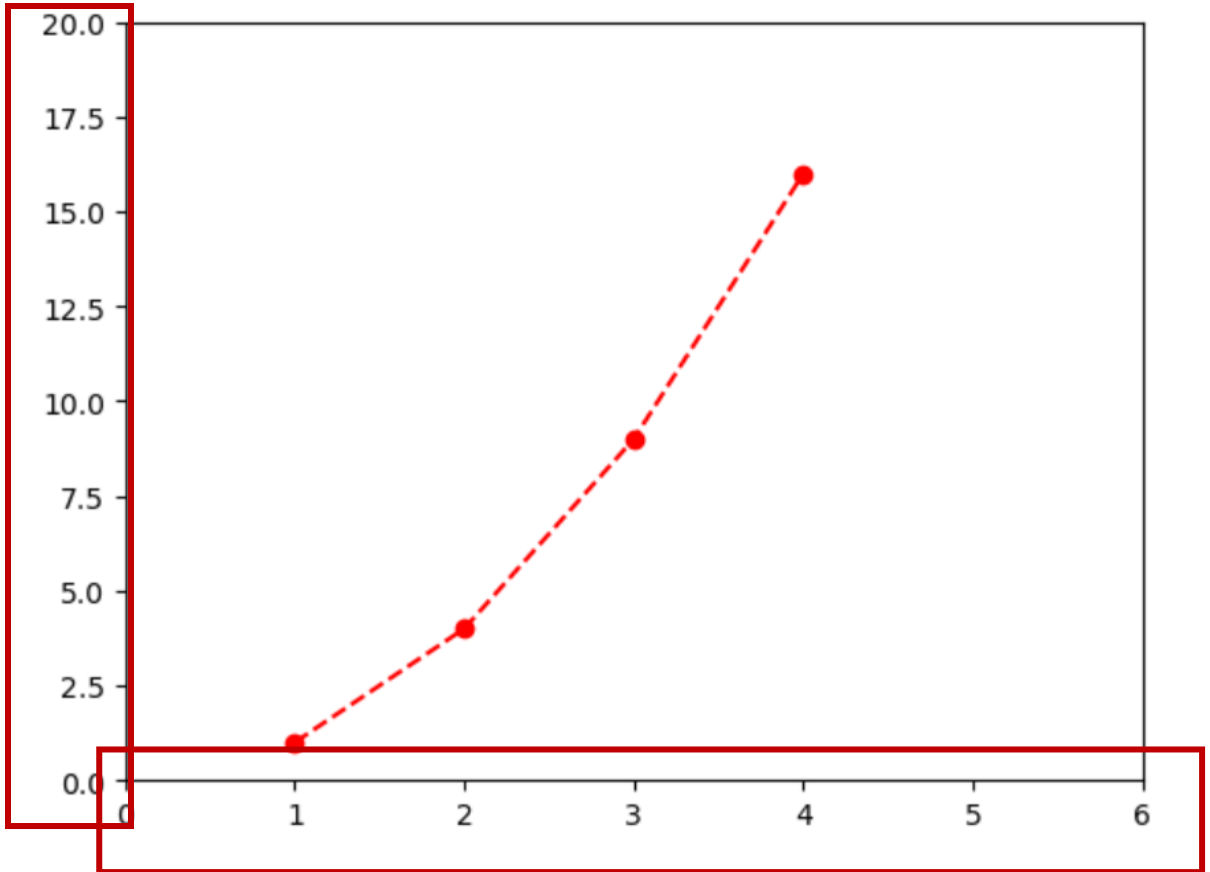
Matplotlib

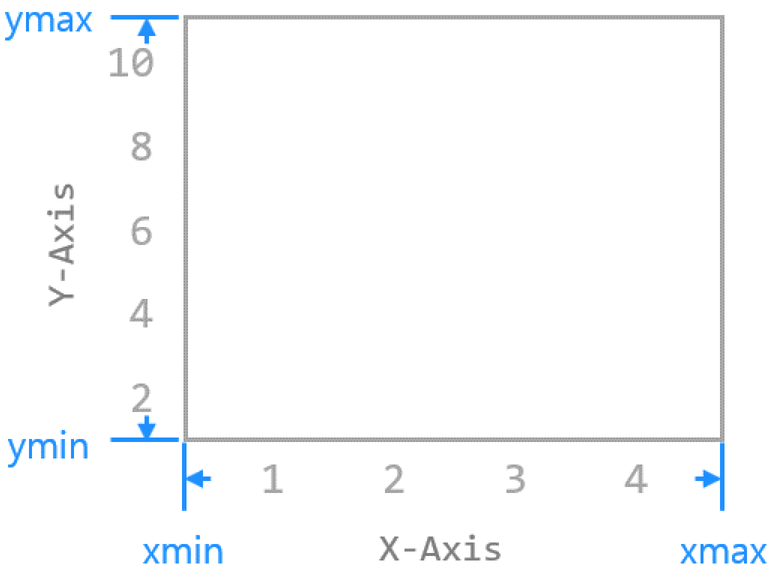
- Line chart : plot()



```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'r--o')
plt.axis((0, 6, 0, 20))
plt.show()
```

Axis limits

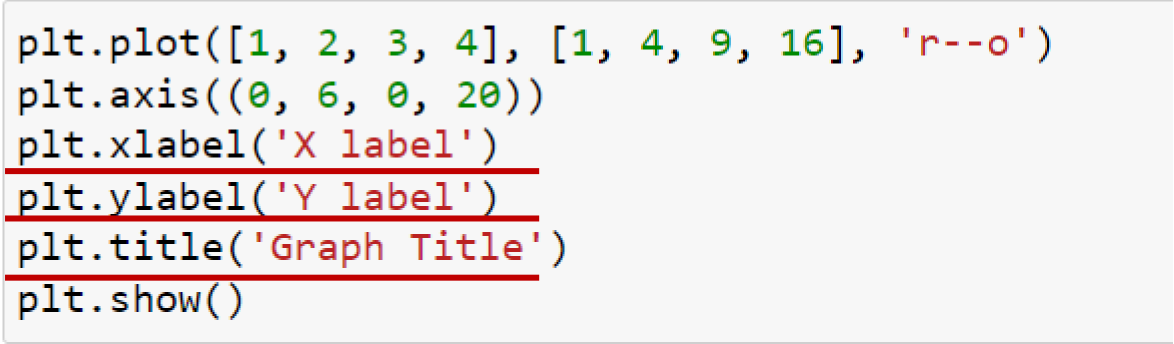




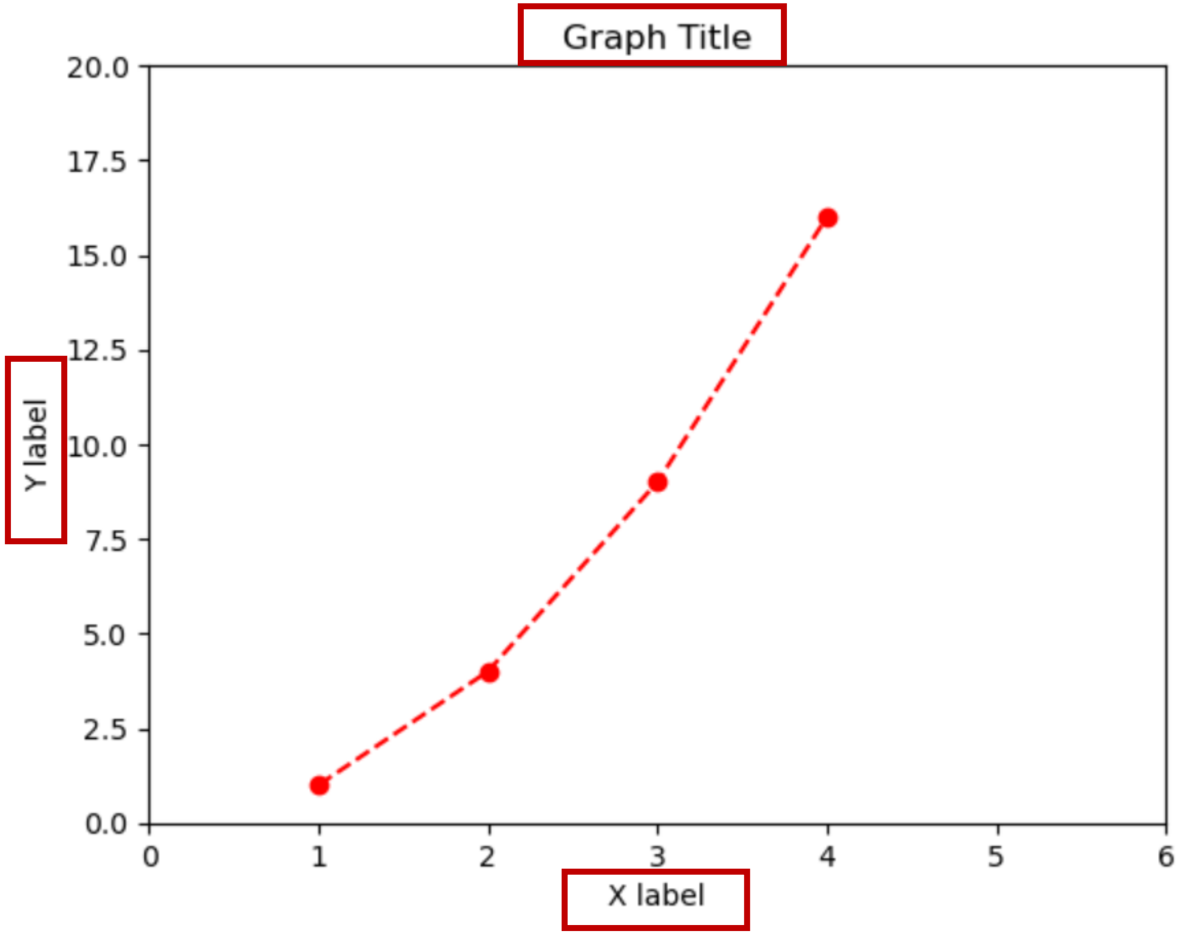
```
plt.axis((xmin, xmax, ymin, ymax))
plt.axis([xmin, xmax, ymin, ymax])
```

Matplotlib

- Line chart : plot()

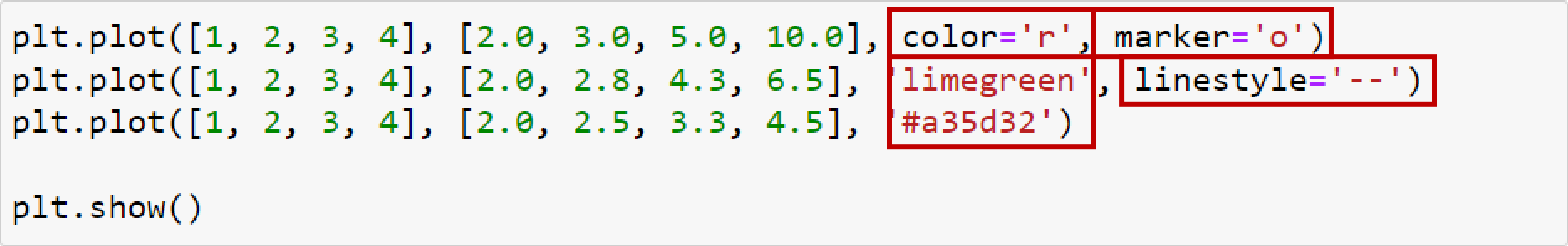


```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'r--o')
plt.axis((0, 6, 0, 20))
plt.xlabel('X label')
plt.ylabel('Y label')
plt.title('Graph Title')
plt.show()
```

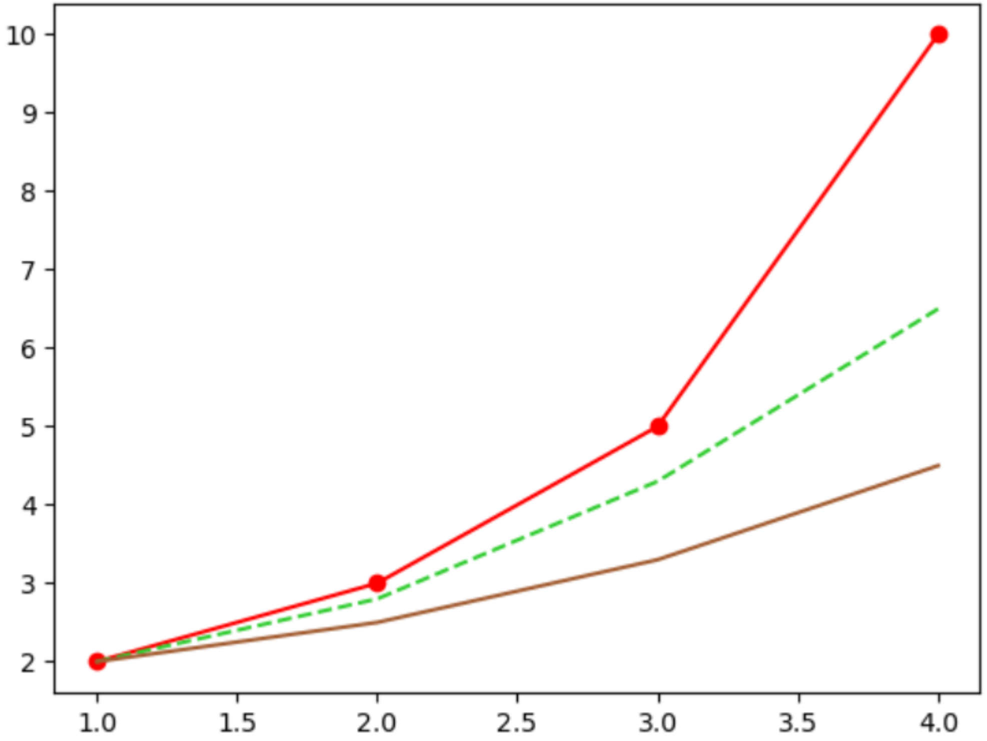


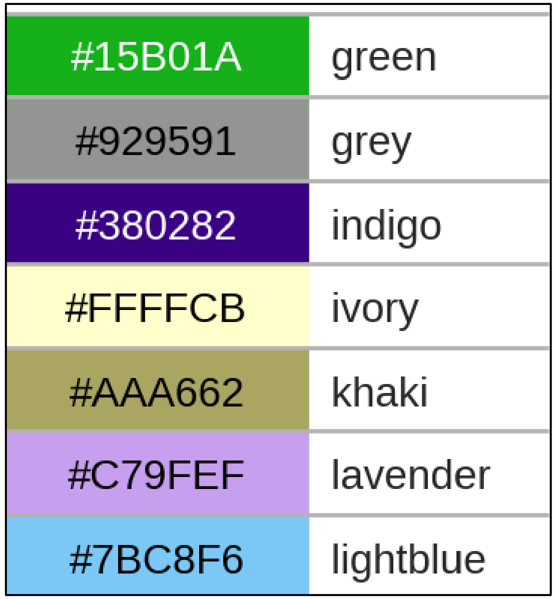
Matplotlib

- Line chart : plot()

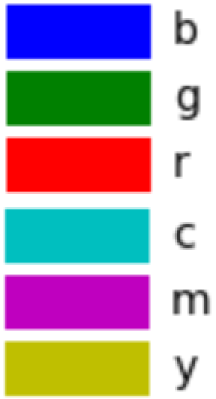


```
plt.plot([1, 2, 3, 4], [2.0, 3.0, 5.0, 10.0], color='r', marker='o')
plt.plot([1, 2, 3, 4], [2.0, 2.8, 4.3, 6.5], 'limegreen', linestyle='--')
plt.plot([1, 2, 3, 4], [2.0, 2.5, 3.3, 4.5], '#a35d32')

plt.show()
```





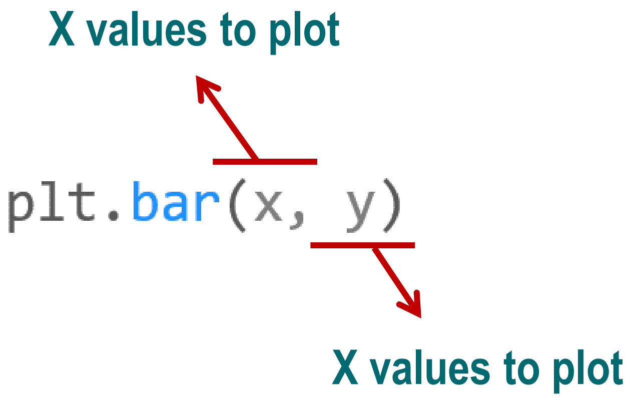
#15B01A	green
#929591	grey
#380282	indigo
#FFFFCB	ivory
#AAA662	khaki
#C79FEF	lavender
#7BC8F6	lightblue



🟦	b
🟩	g
🟥	r
■	c
🟪	m
🟨	y

Matplotlib

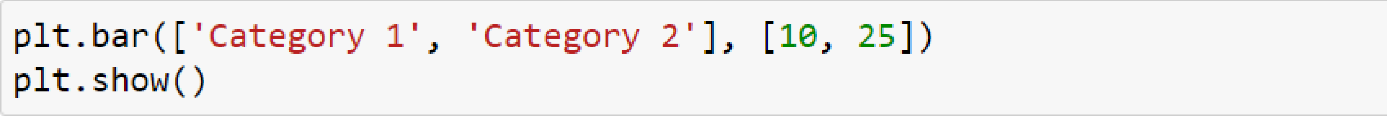
- Bar chart: bar()



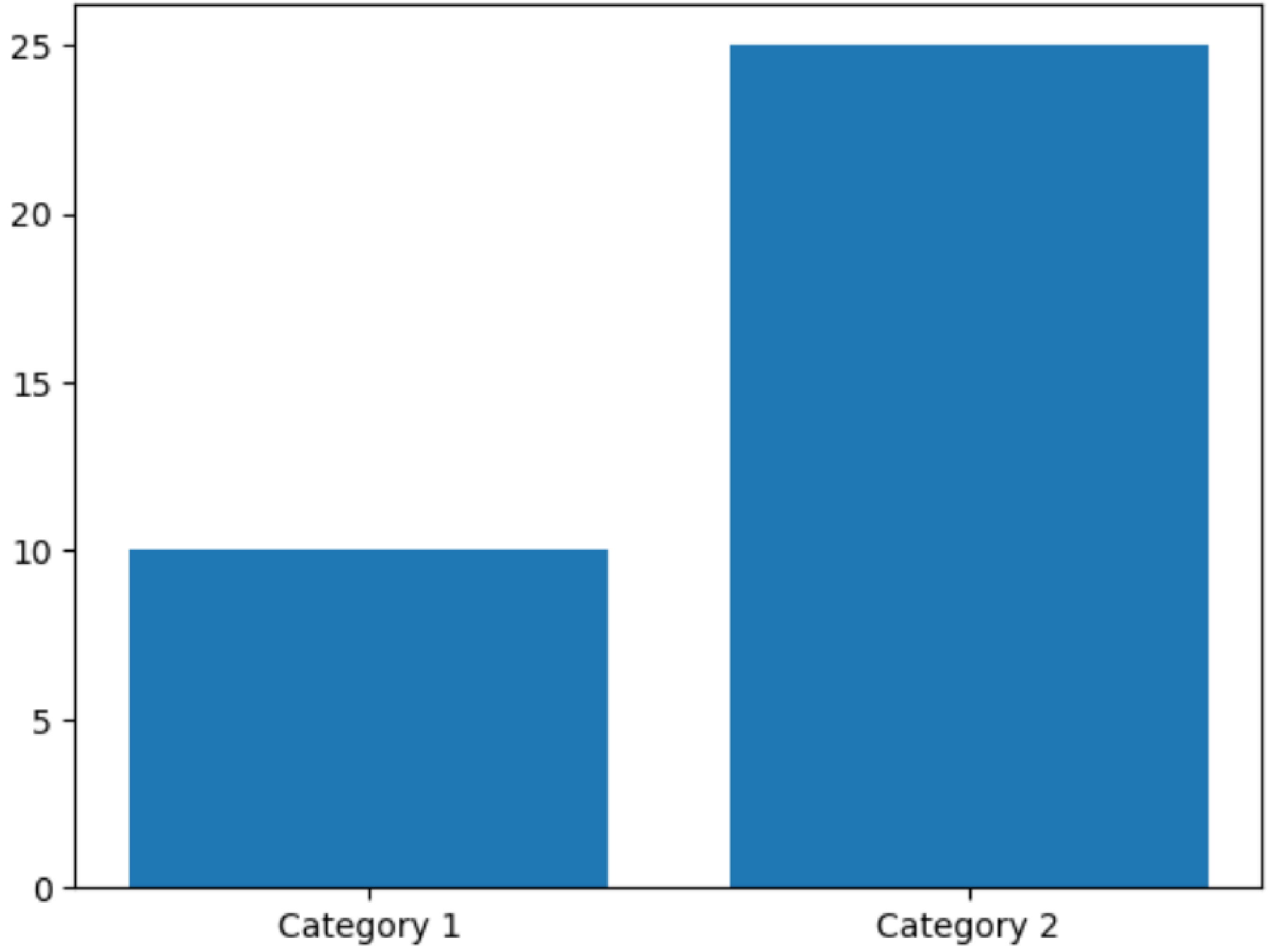
X values to plot

```
plt.bar(x, y)
```

X values to plot

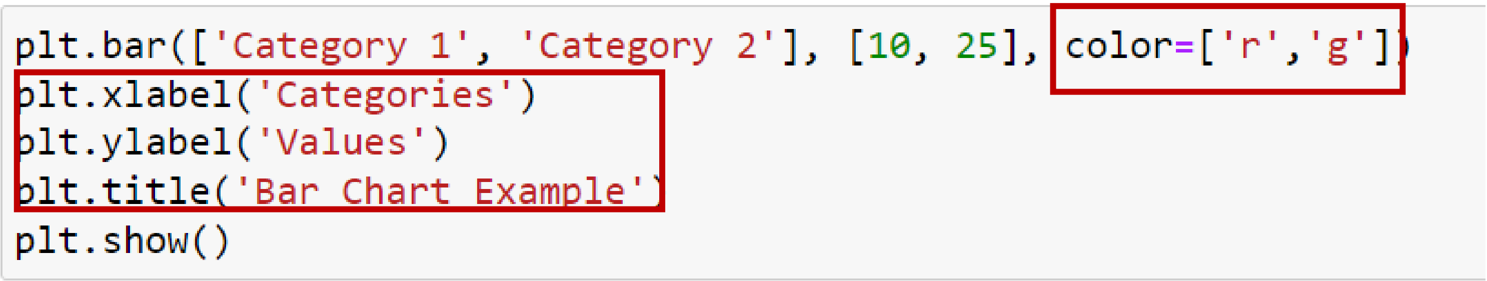


```
plt.bar(['Category 1', 'Category 2'], [10, 25])
plt.show()
```

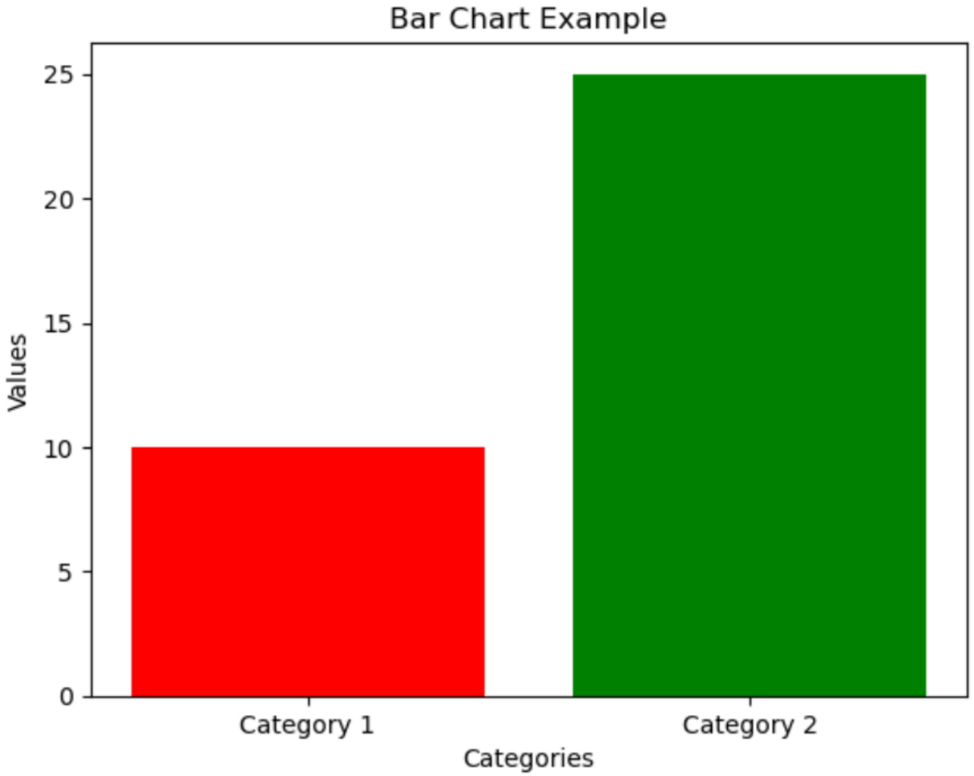


Matplotlib

- **Bar chart: bar()**

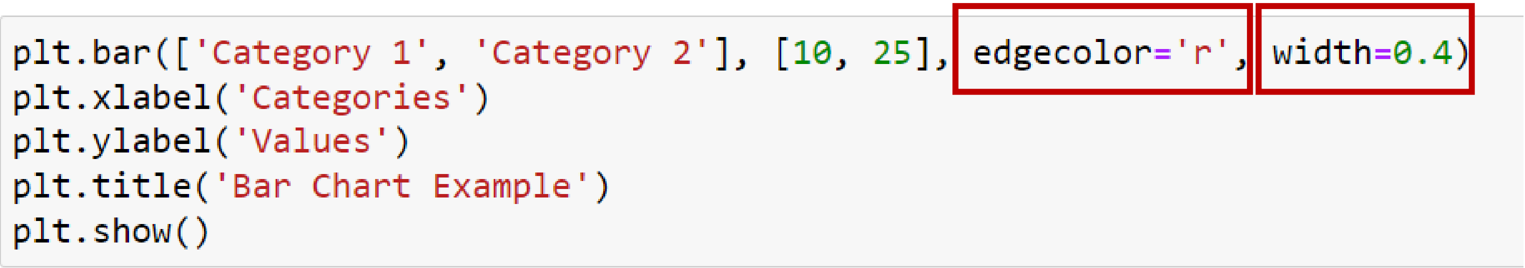


```
plt.bar(['Category 1', 'Category 2'], [10, 25], color=['r','g'])
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Bar Chart Example')
plt.show()
```

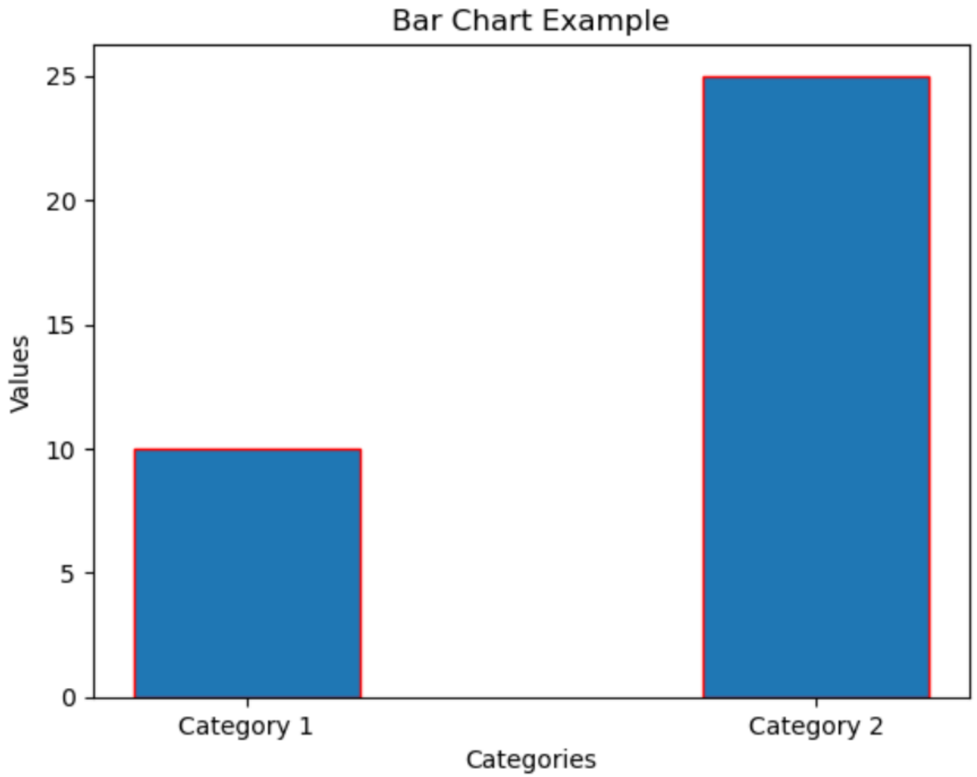


Matplotlib

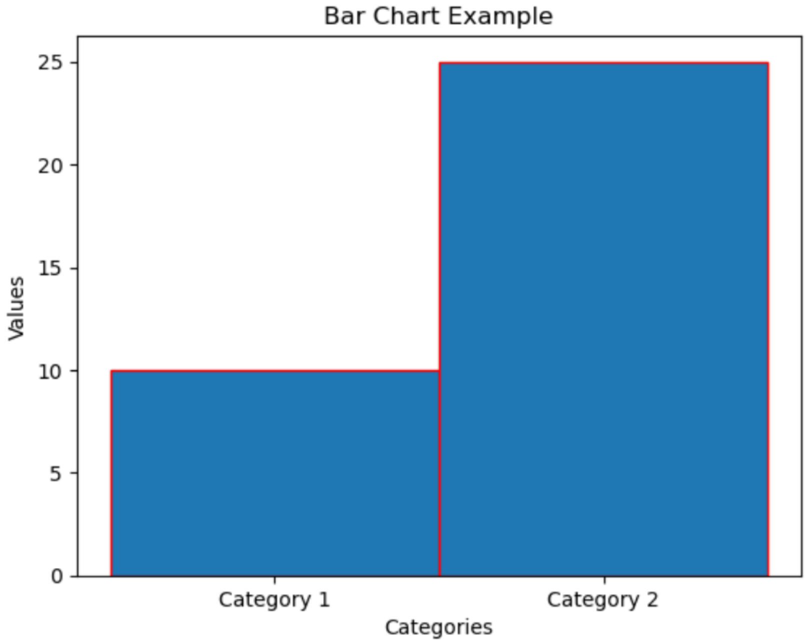
- Bar chart: bar()



```
plt.bar(['Category 1', 'Category 2'], [10, 25], edgecolor='r', width=0.4)
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Bar Chart Example')
plt.show()
```

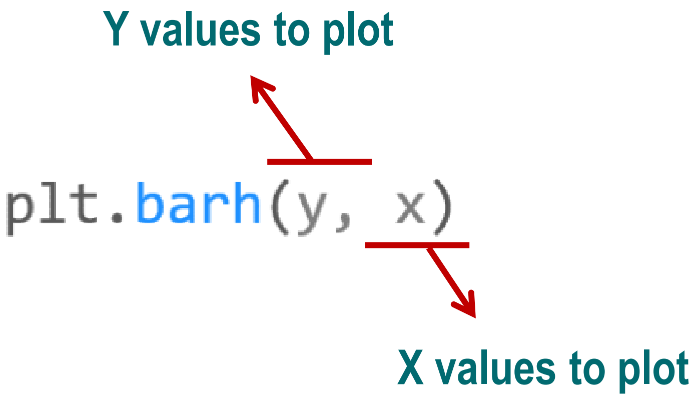


width=1.0 :



Matplotlib

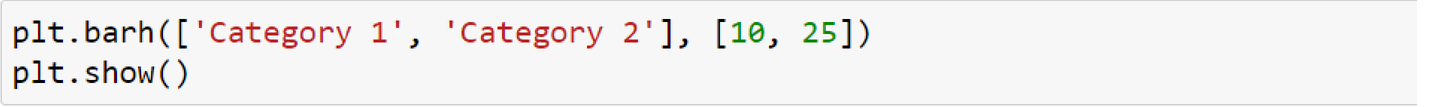
- **Bar chart: barh()**



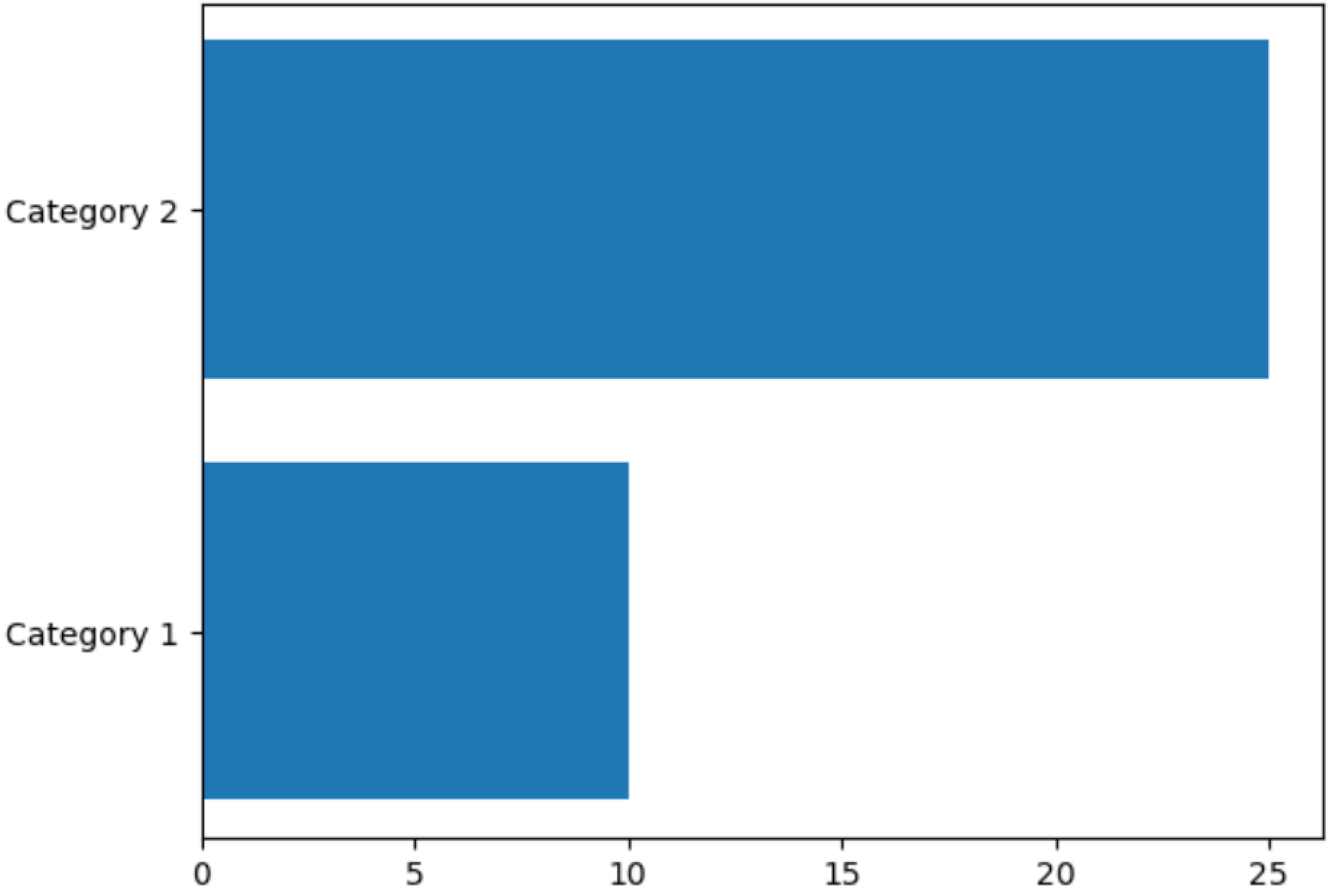
Y values to plot

```
plt.barh(y, x)
```

X values to plot

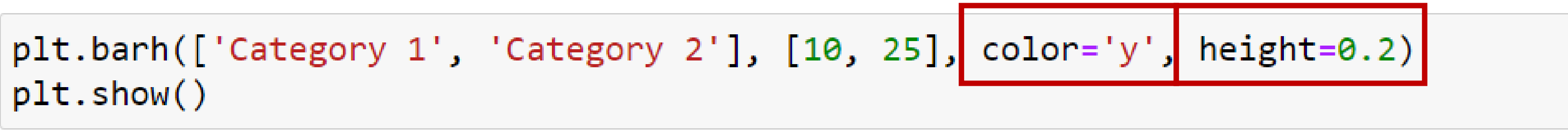


```
plt.barh(['Category 1', 'Category 2'], [10, 25])
plt.show()
```

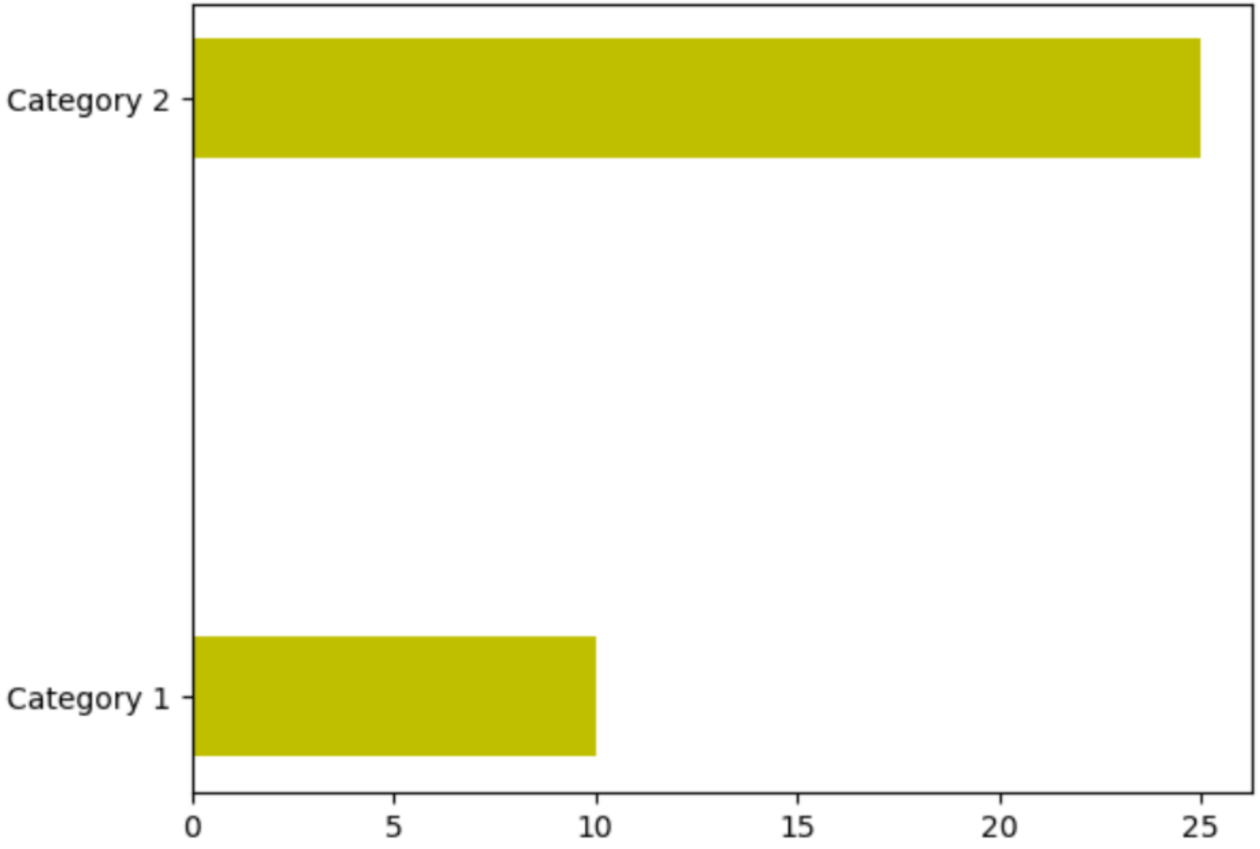


Matplotlib

- Bar chart: barh()

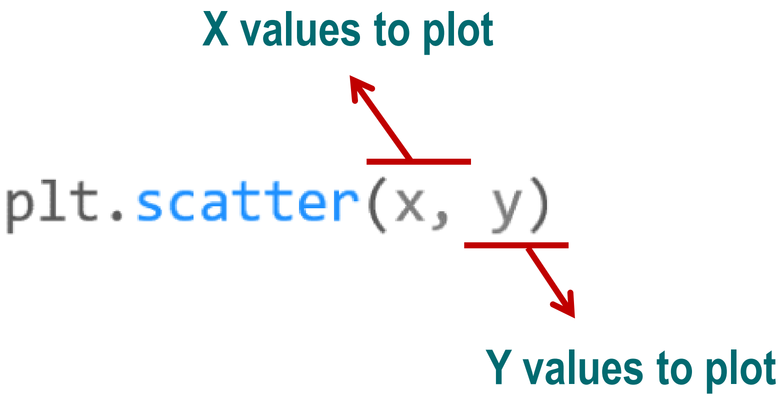


```
plt.barh(['Category 1', 'Category 2'], [10, 25], color='y', height=0.2)
plt.show()
```



Matplotlib

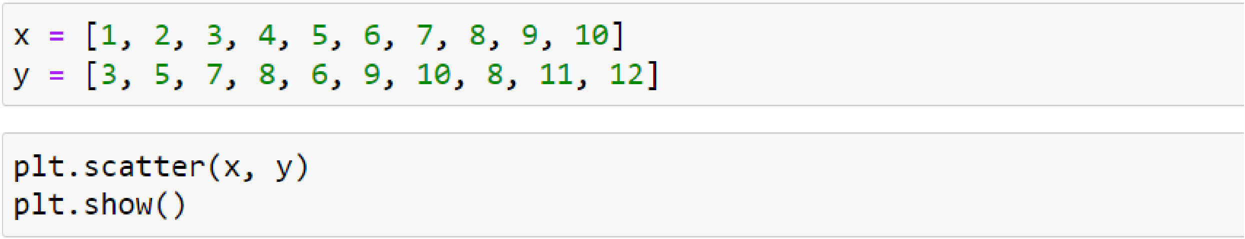
- Scatter plot: scatter()



X values to plot

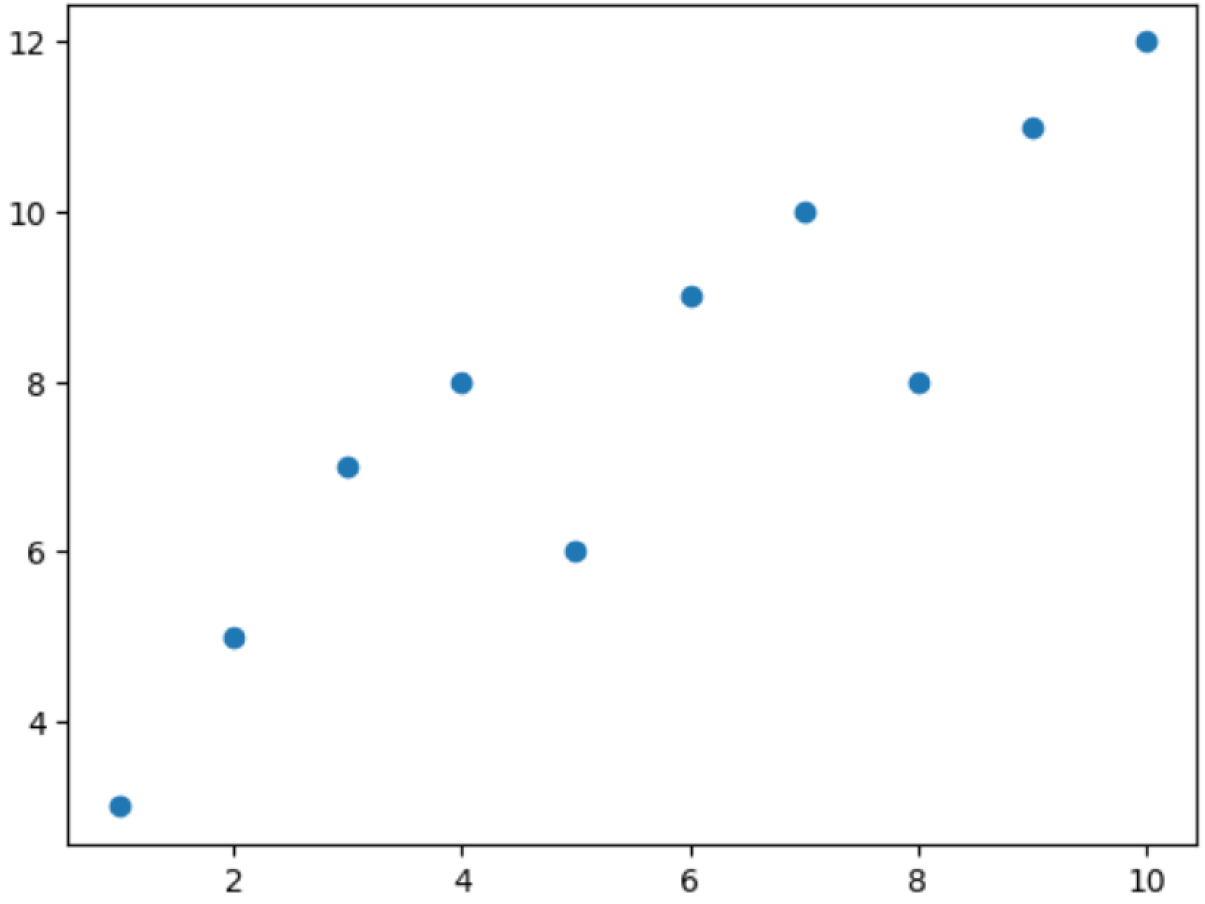
```
plt.scatter(x, y)
```

Y values to plot



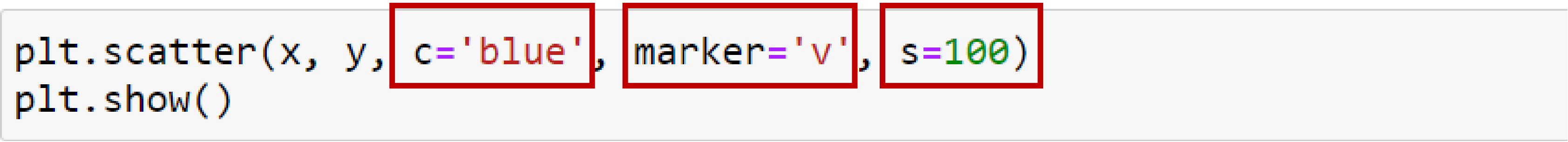
```
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [3, 5, 7, 8, 6, 9, 10, 8, 11, 12]
```

```
plt.scatter(x, y)
plt.show()
```

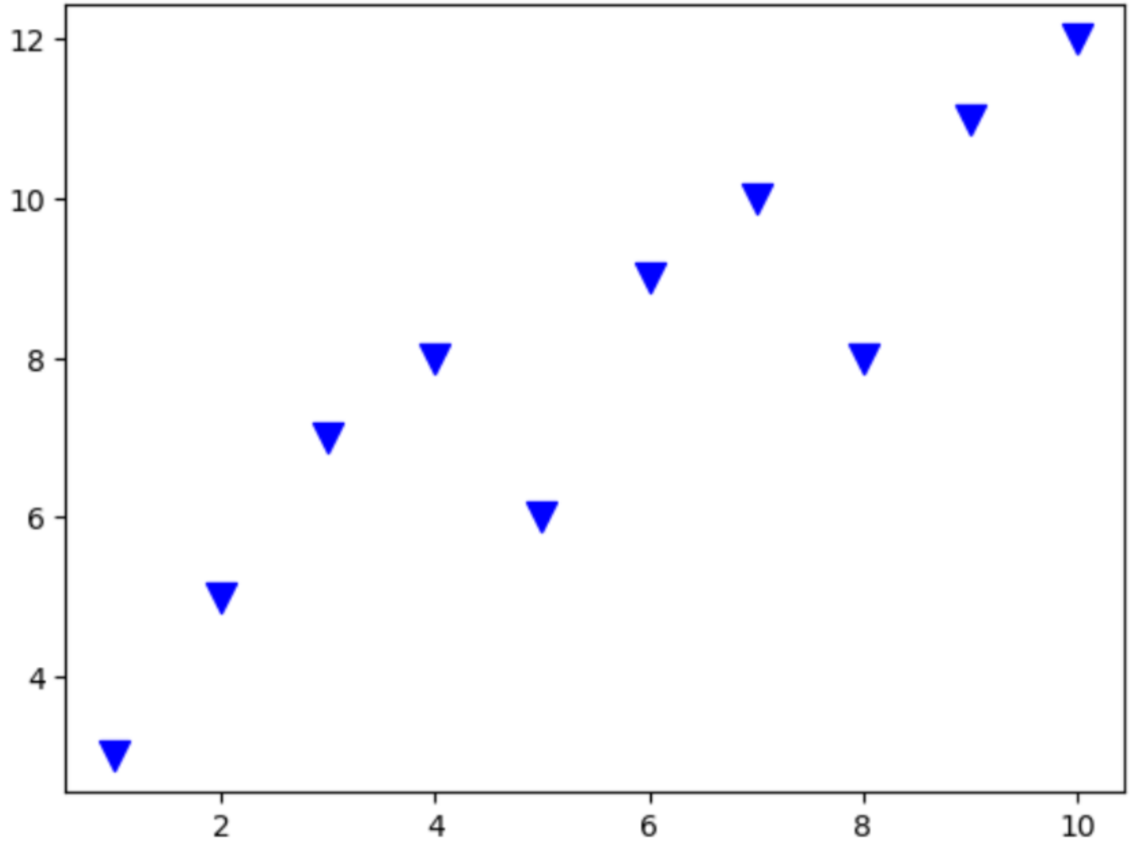


Matplotlib

- **Scatter plot: scatter()**

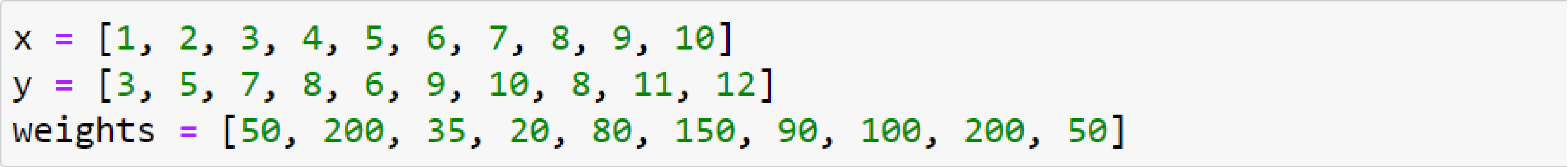


```
plt.scatter(x, y, c='blue', marker='v', s=100)
plt.show()
```

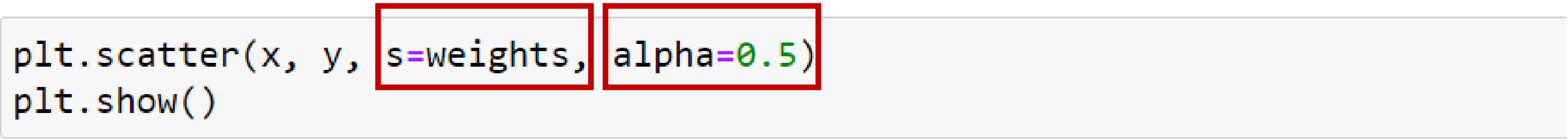


Matplotlib

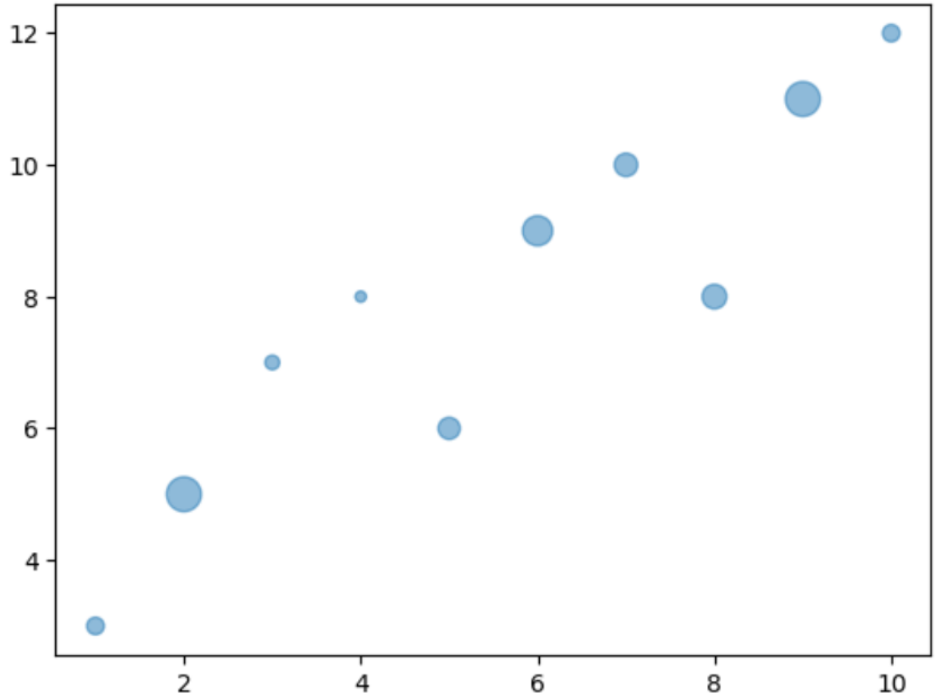
- ## Scatter plot: scatter()



```
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [3, 5, 7, 8, 6, 9, 10, 8, 11, 12]
weights = [50, 200, 35, 20, 80, 150, 90, 100, 200, 50]
```



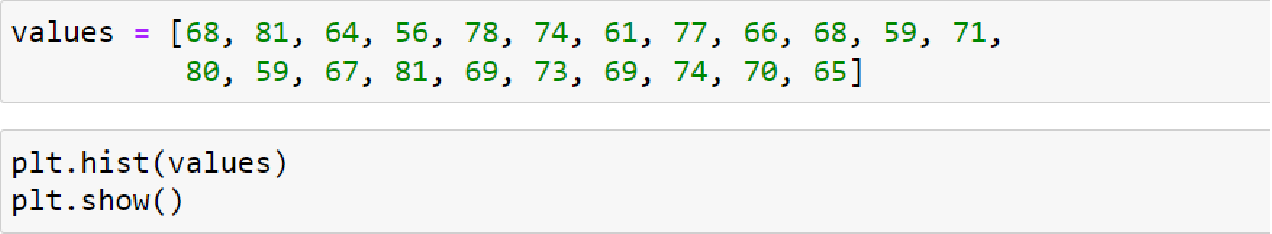
```
plt.scatter(x, y, s=weights, alpha=0.5)
plt.show()
```



Matplotlib

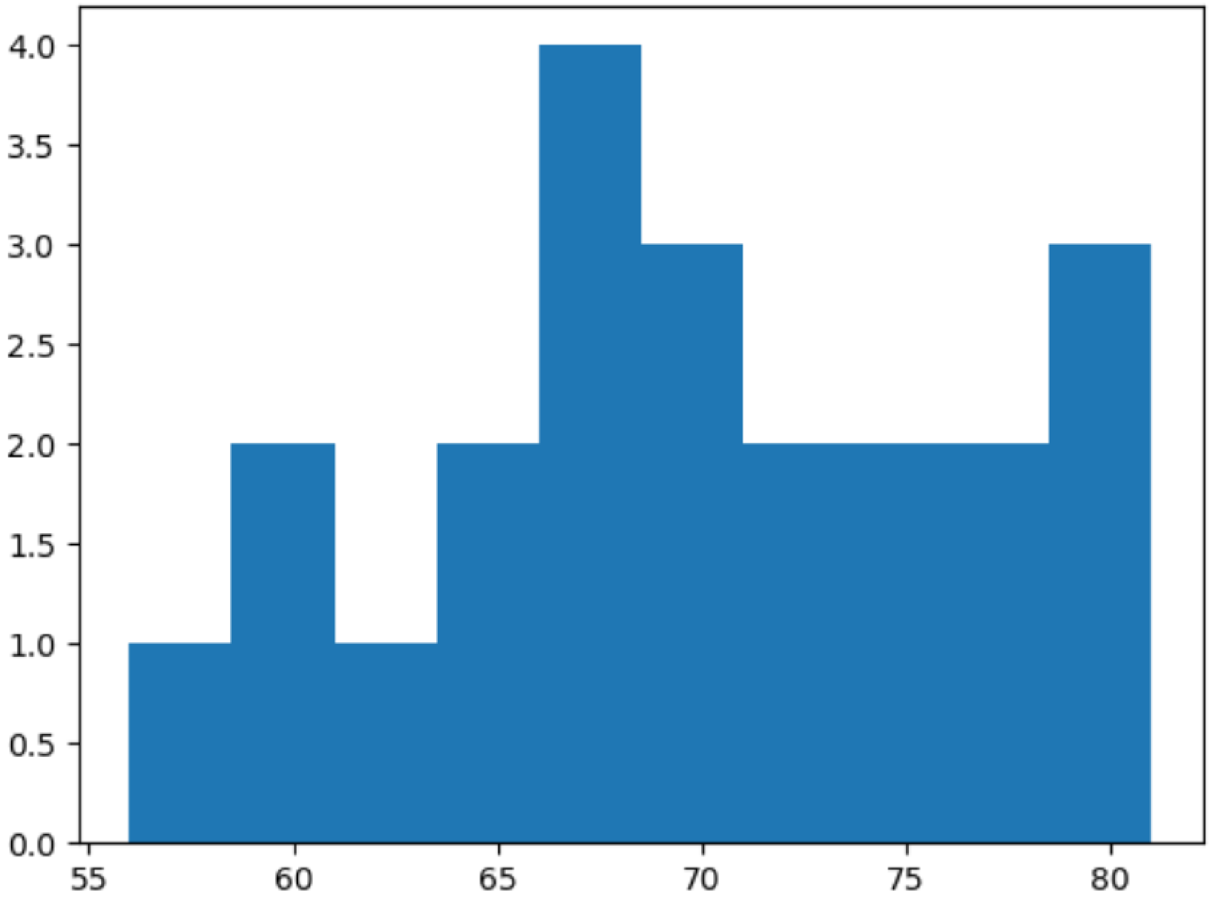
- Histogram: hist()

```
plt.hist(values)
```



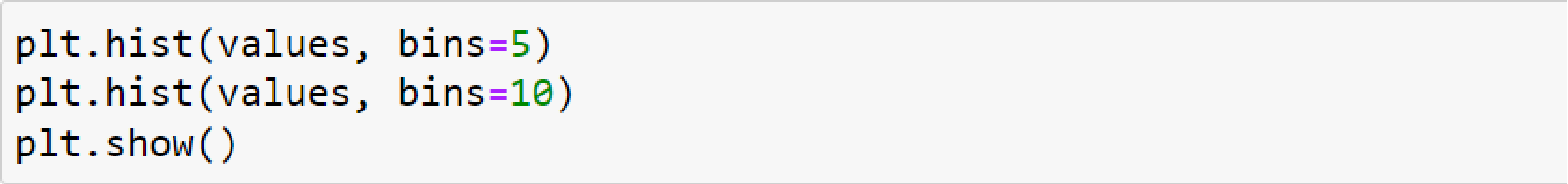
```
values = [68, 81, 64, 56, 78, 74, 61, 77, 66, 68, 59, 71,
          80, 59, 67, 81, 69, 73, 69, 74, 70, 65]
```

```
plt.hist(values)
plt.show()
```

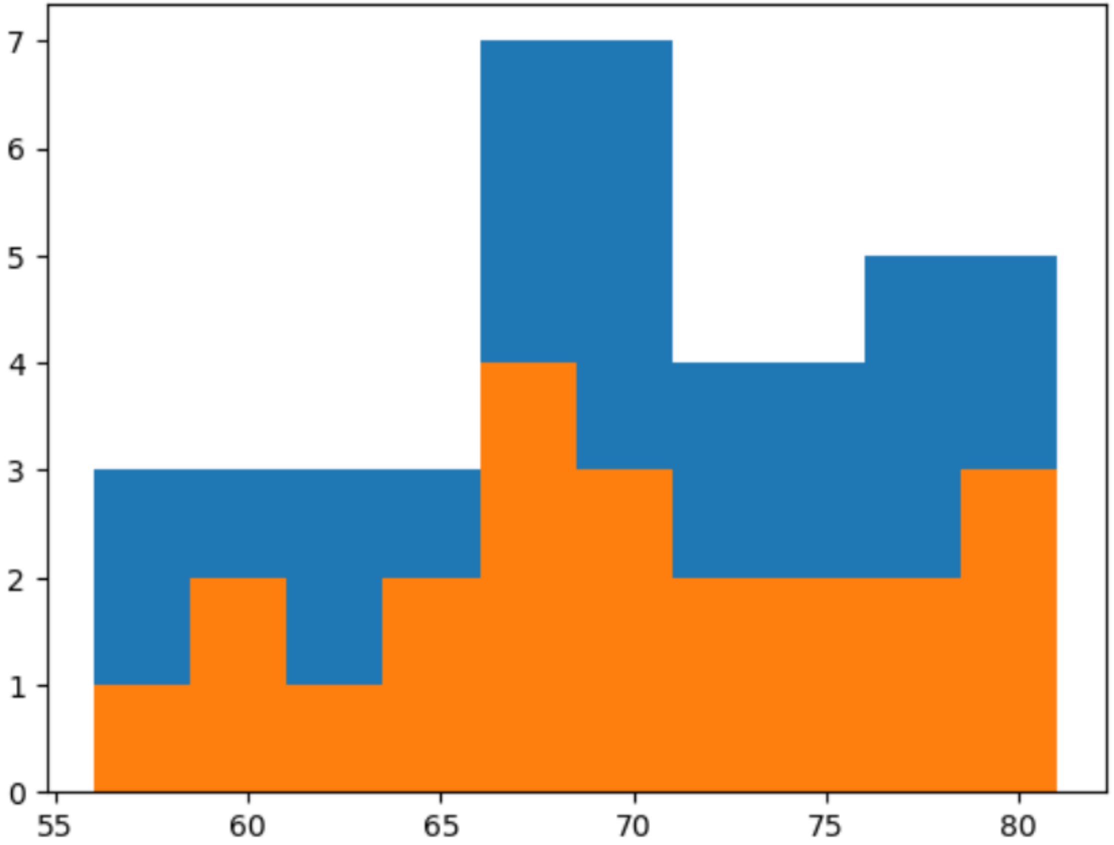


Matplotlib

- **Histogram: hist()**

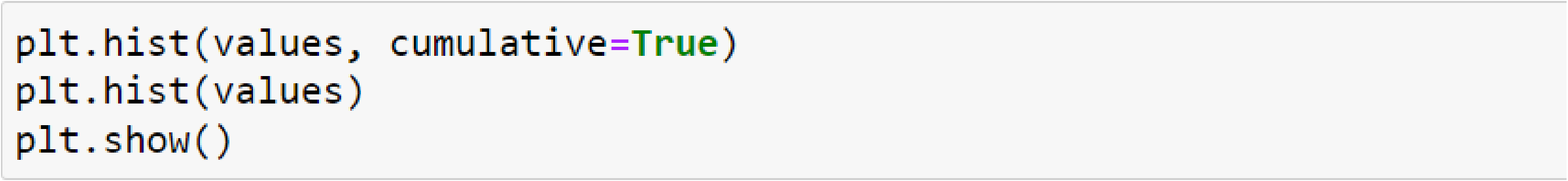


```
plt.hist(values, bins=5)
plt.hist(values, bins=10)
plt.show()
```

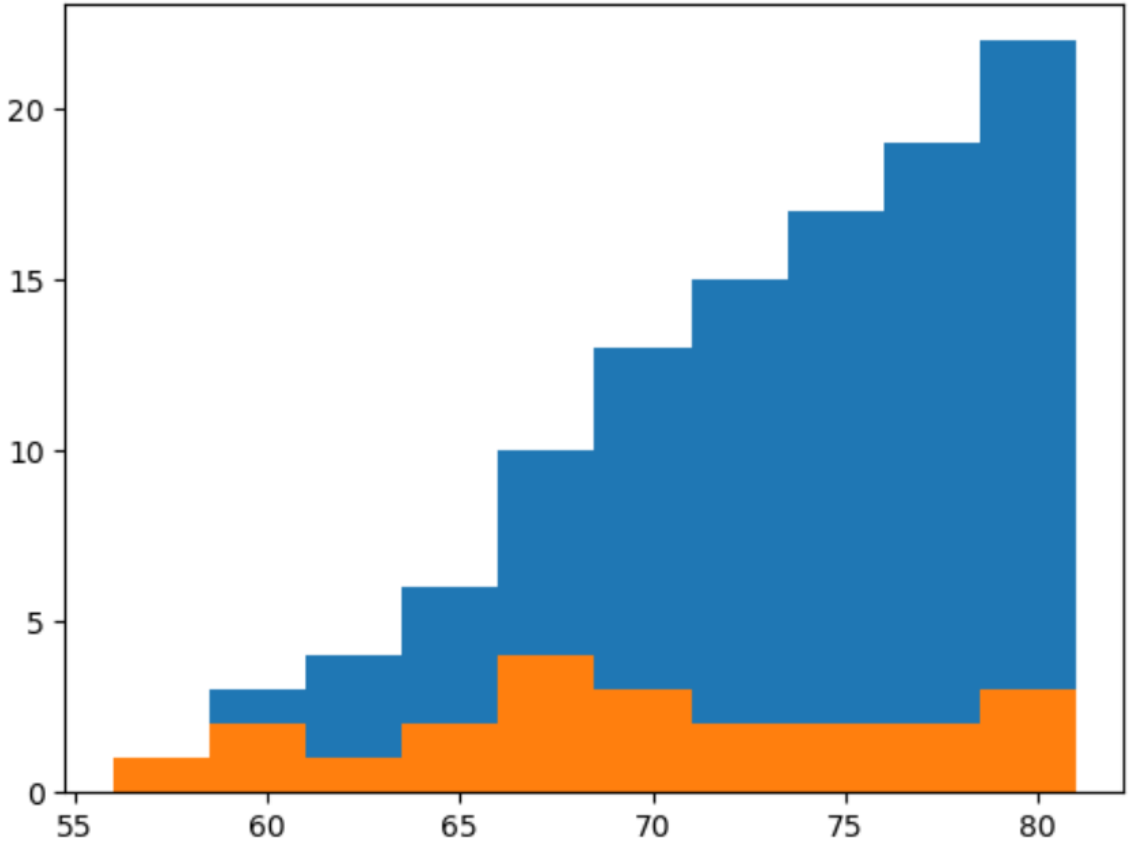


Matplotlib

- Histogram: hist()



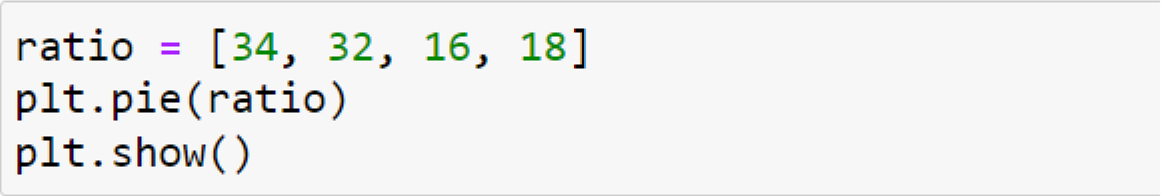
```
plt.hist(values, cumulative=True)
plt.hist(values)
plt.show()
```



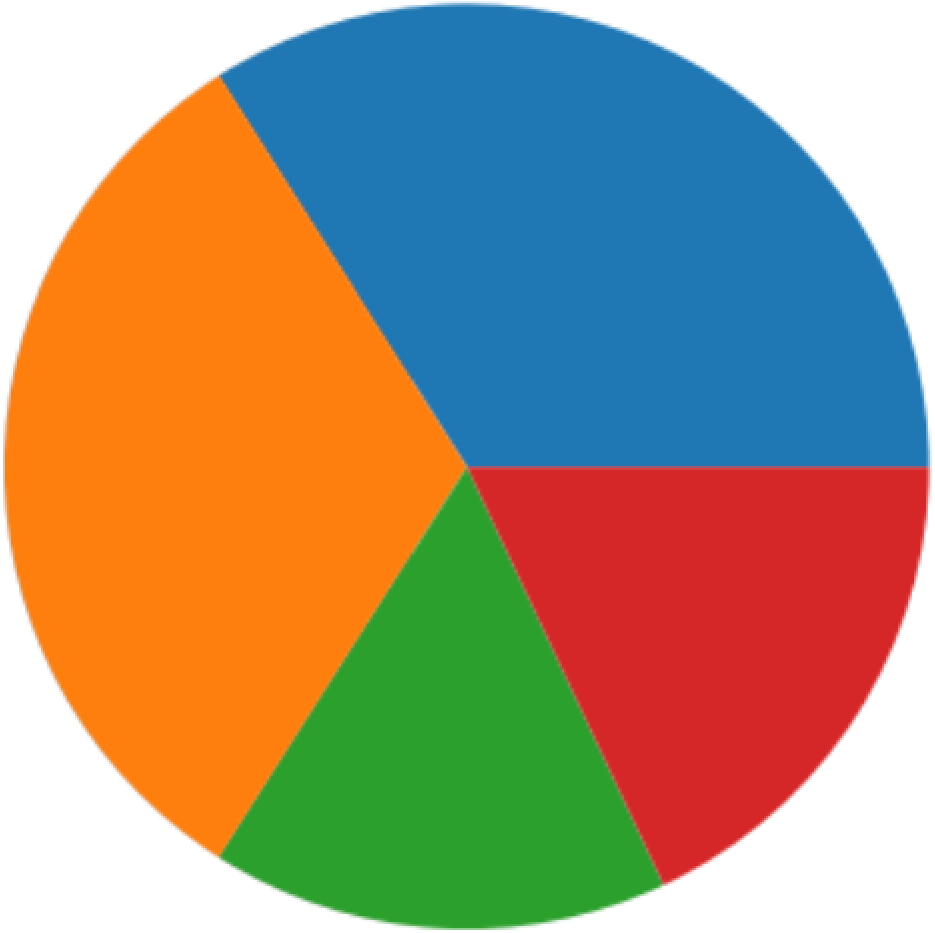
Matplotlib

- **Pie chart: pie()**

```
plt.pie(values)
```

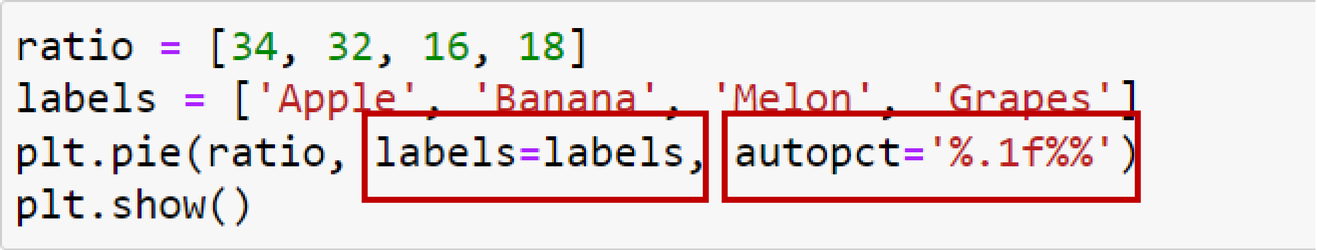


```
ratio = [34, 32, 16, 18]
plt.pie(ratio)
plt.show()
```

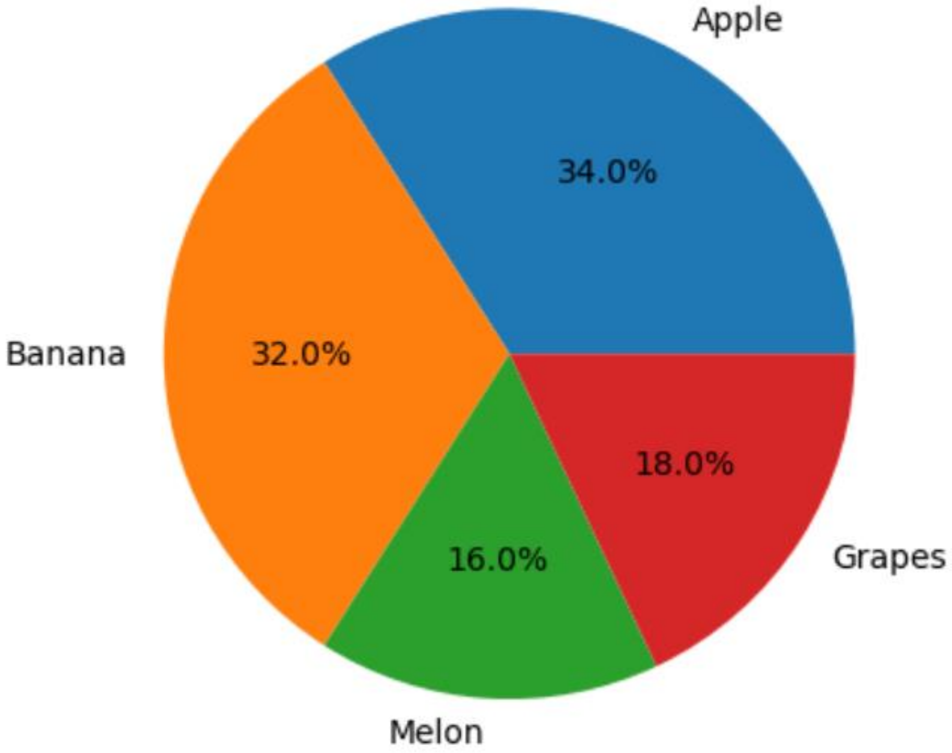


Matplotlib

- **Pie chart: pie()**



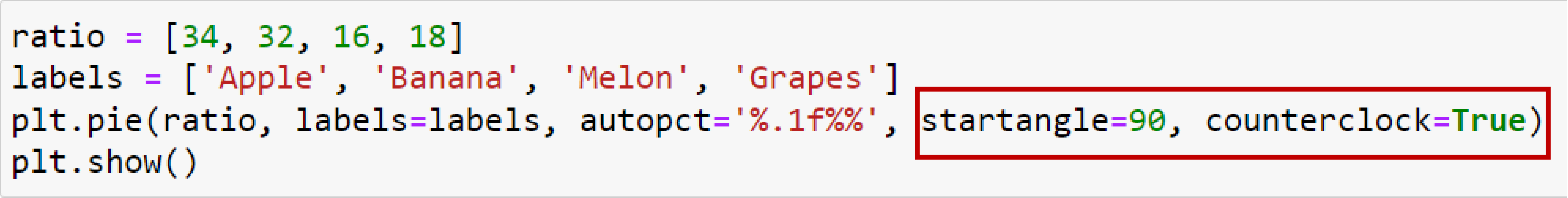
```
ratio = [34, 32, 16, 18]
labels = ['Apple', 'Banana', 'Melon', 'Grapes']
plt.pie(ratio, labels=labels, autopct='%.1f%%')
plt.show()
```



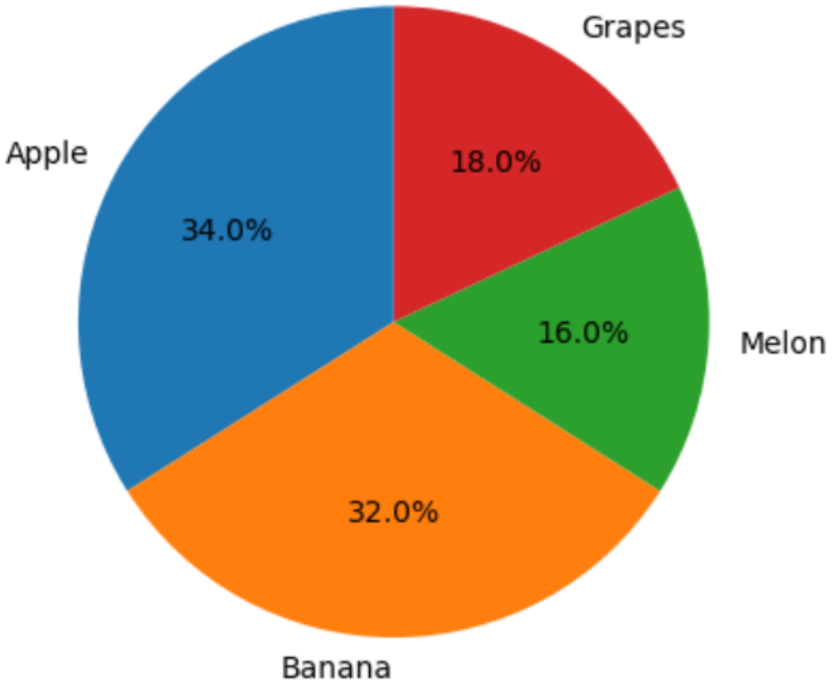
Format string	Output using ‘45.793’
autopct=‘%f’	45.793
autopct=‘%.1f’	45.8
autopct=‘%.2f’	45.79
autopct=‘%.1f%%’	45.8%
autopct=‘p=%.1f%%’	p=45.8%

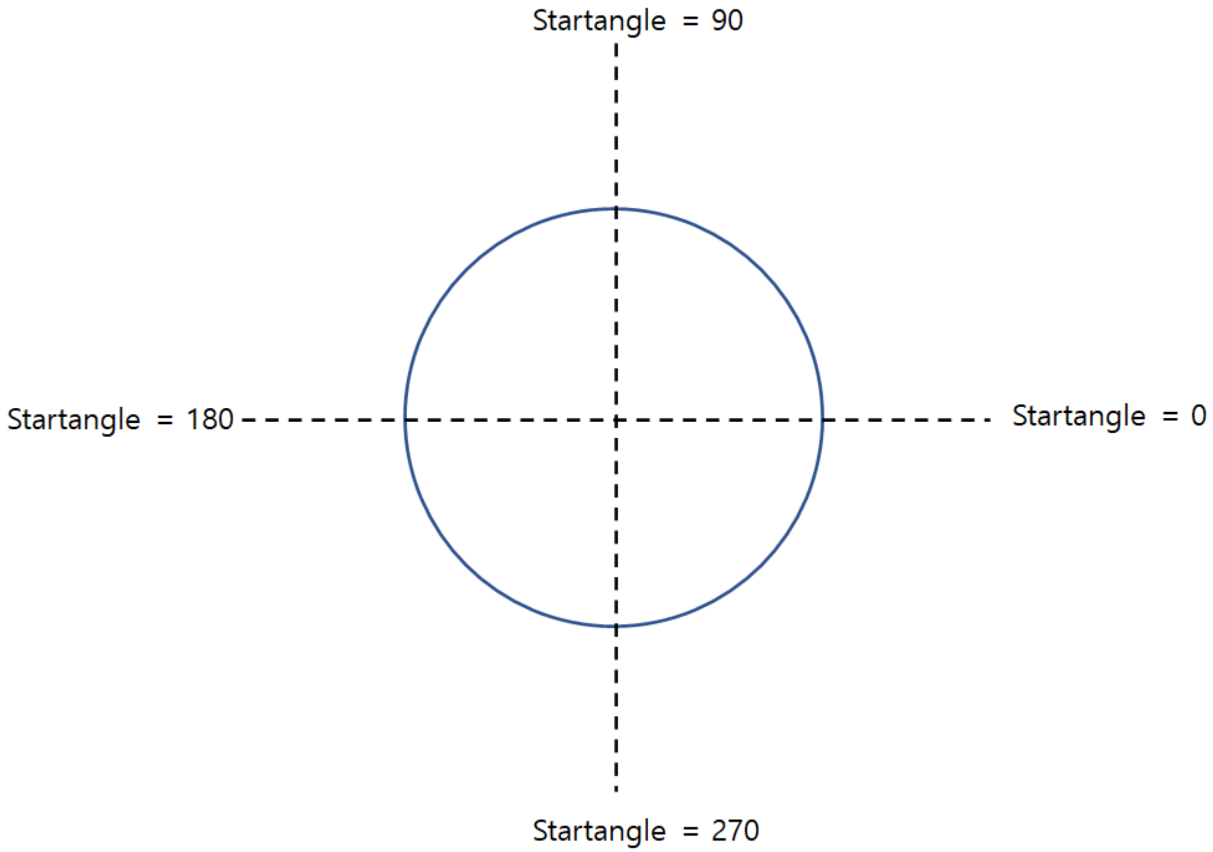
Matplotlib

- **Pie chart: pie()**



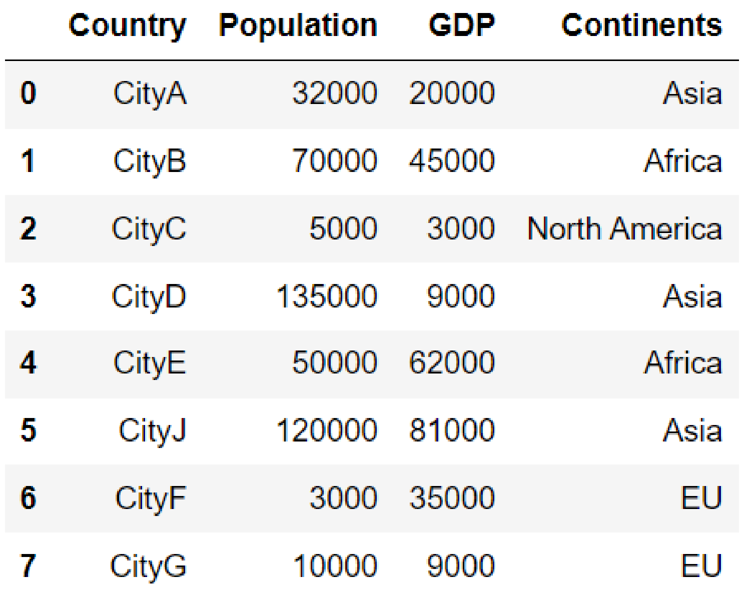
```
ratio = [34, 32, 16, 18]
labels = ['Apple', 'Banana', 'Melon', 'Grapes']
plt.pie(ratio, labels=labels, autopct='%.1f%%', startangle=90, counterclock=True)
plt.show()
```



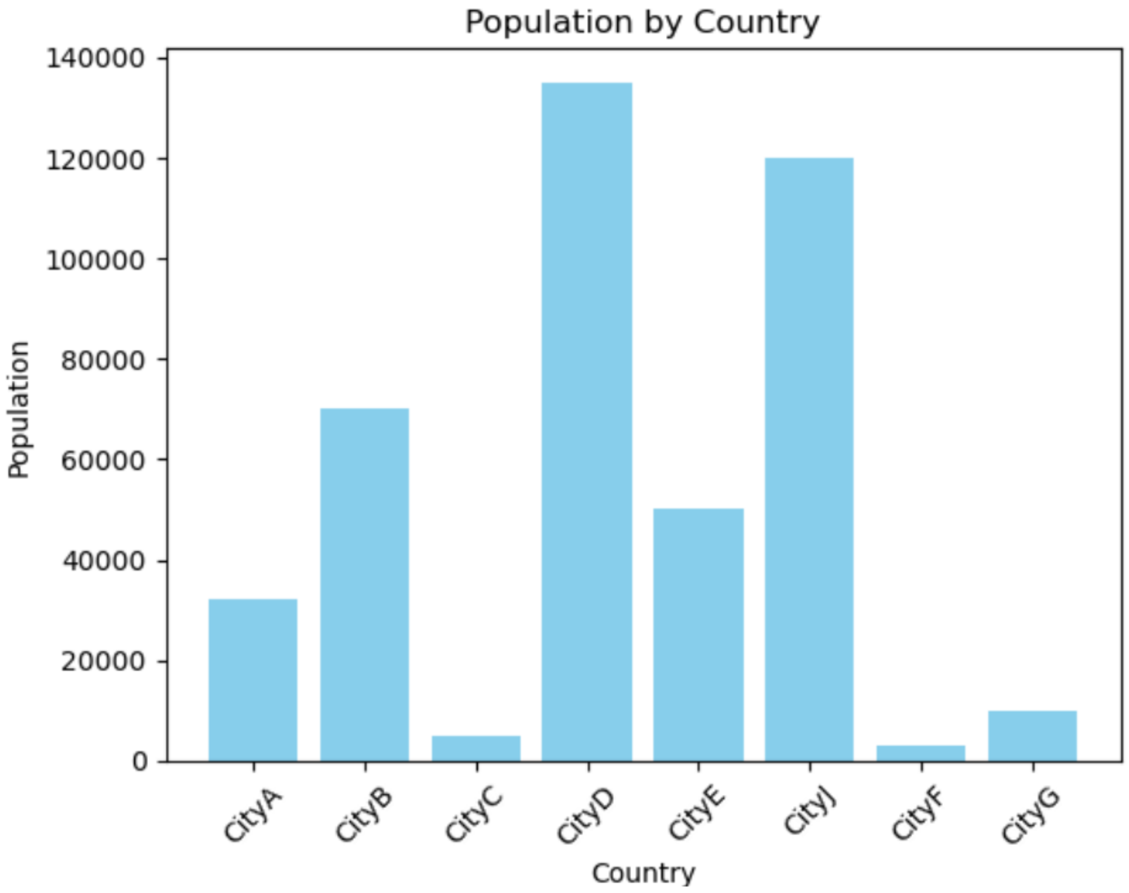


Matplotlib

- **Practice: Population by Country**

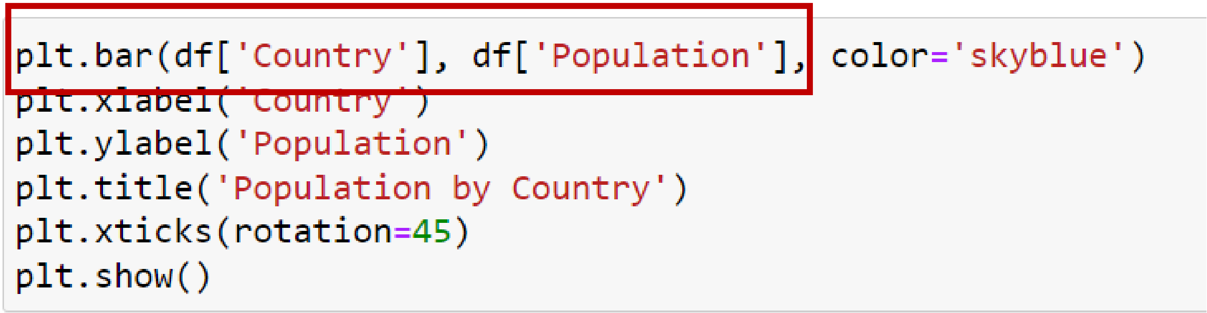


	Country	Population	GDP	Continents
0	CityA	32000	20000	Asia
1	CityB	70000	45000	Africa
2	CityC	5000	3000	North America
3	CityD	135000	9000	Asia
4	CityE	50000	62000	Africa
5	CityJ	120000	81000	Asia
6	CityF	3000	35000	EU
7	CityG	10000	9000	EU

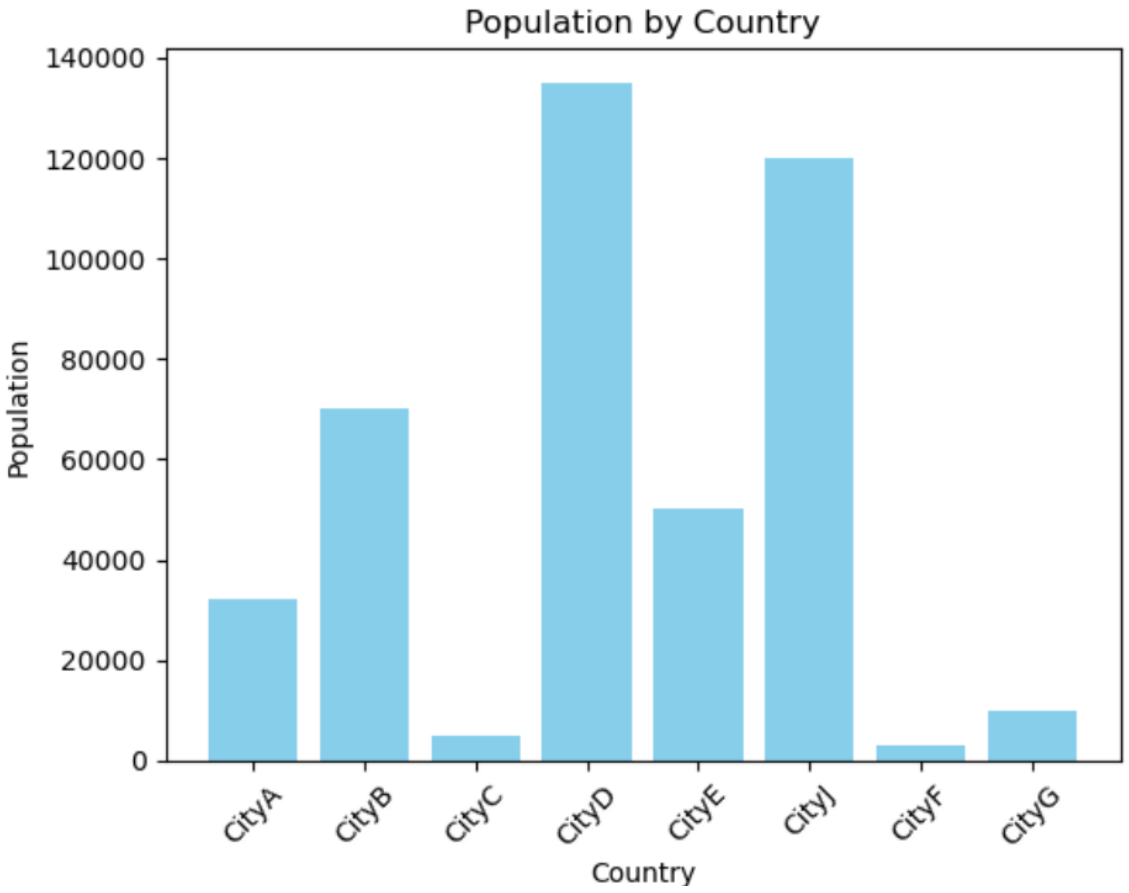


Matplotlib

- Practice: Population by Country

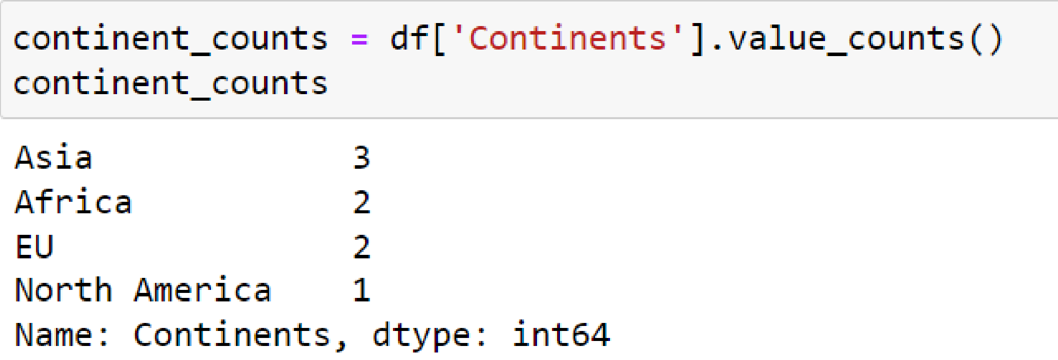


```
plt.bar(df['Country'], df['Population'], color='skyblue')
plt.xlabel('Country')
plt.ylabel('Population')
plt.title('Population by Country')
plt.xticks(rotation=45)
plt.show()
```



Matplotlib

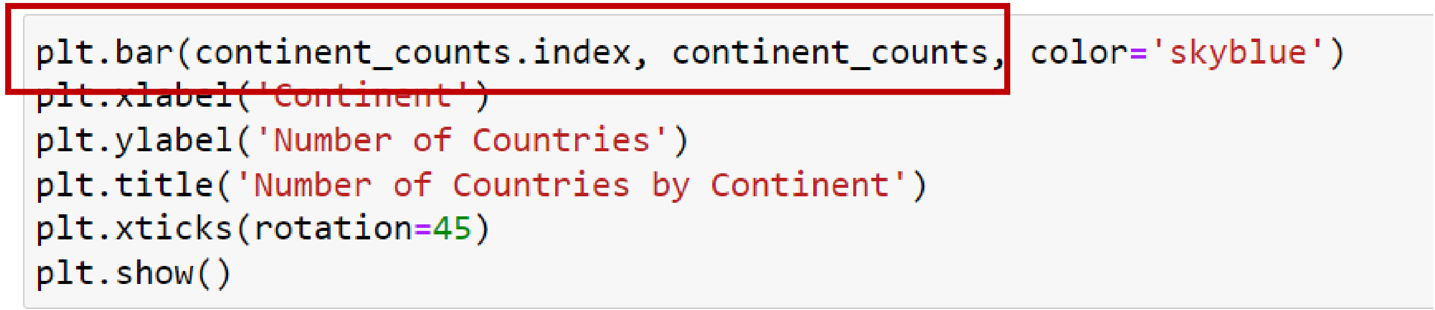
- **Practice: Number of Countries by Continent**
  - Calculate the number of countries by continent



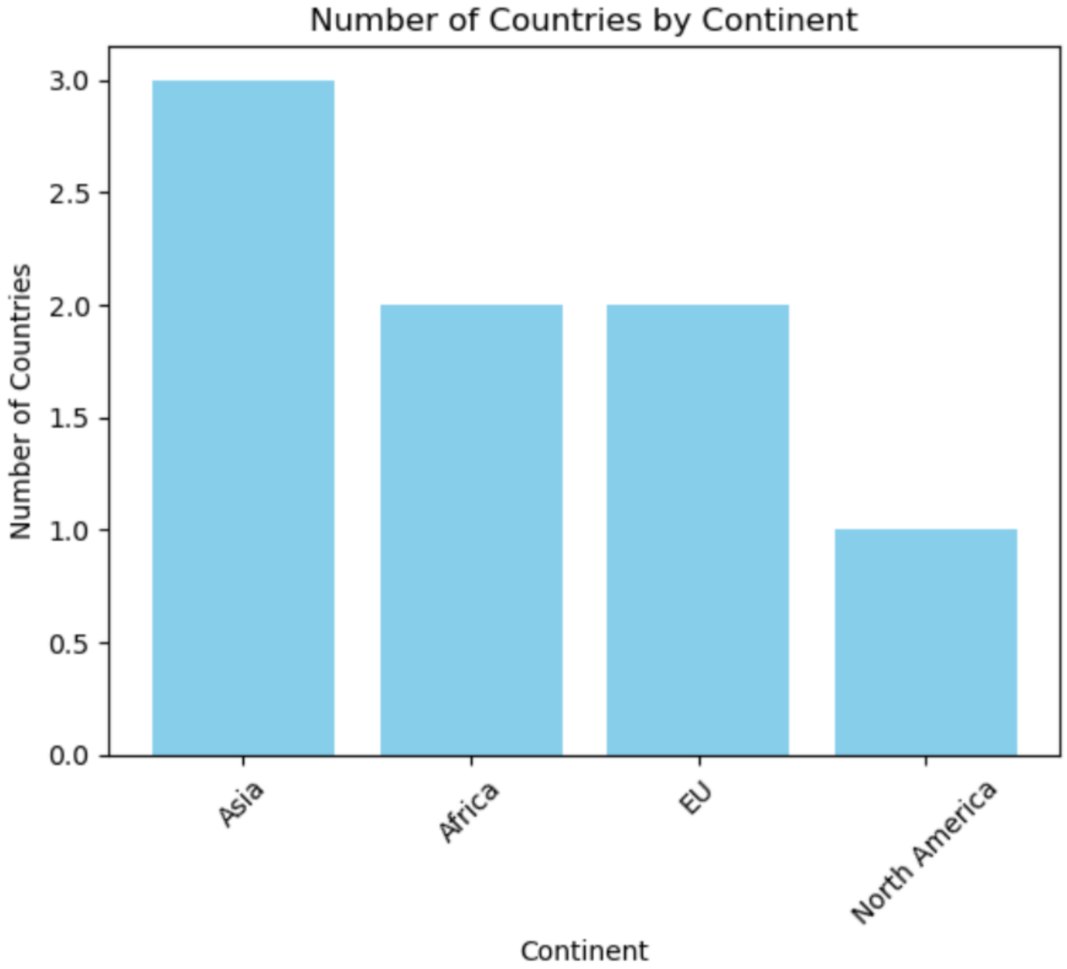
```
continent_counts = df['Continents'].value_counts()
continent_counts
```

```
Asia             3
Africa           2
EU               2
North America    1
Name: Continents, dtype: int64
```

  - Plot the graph



```
plt.bar(continent_counts.index, continent_counts, color='skyblue')
plt.xlabel('Continent')
plt.ylabel('Number of Countries')
plt.title('Number of Countries by Continent')
plt.xticks(rotation=45)
plt.show()
```



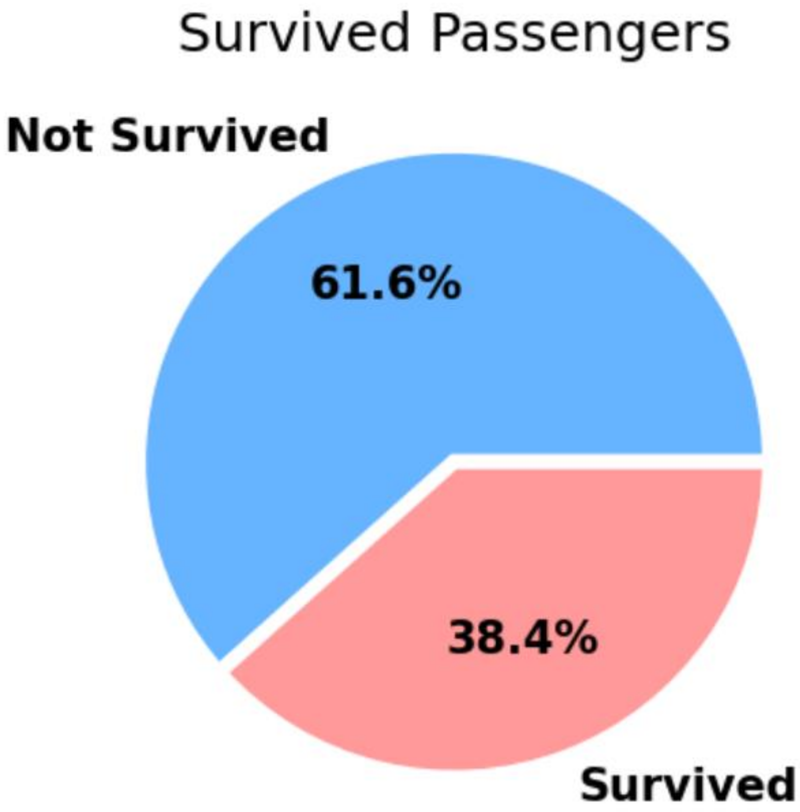
Titanic Dataset Analysis with Visualization

- **Titanic dataset**
  - Titanic dataset includes:
    - Passenger ID
    - Passenger Class (1st, 2nd, or 3rd class)
    - Name
    - Sex
    - Age
    - Sibling/Spouse Aboard (SibSp)
    - Parent/Child Aboard (Parch)
    - Ticket Number
    - Fare
    - Cabin Number
    - Port of Embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)
    - Whether the passenger survived (1 for survived, 0 for did not survive)

Q. What characteristics are likely to have impact on survival?

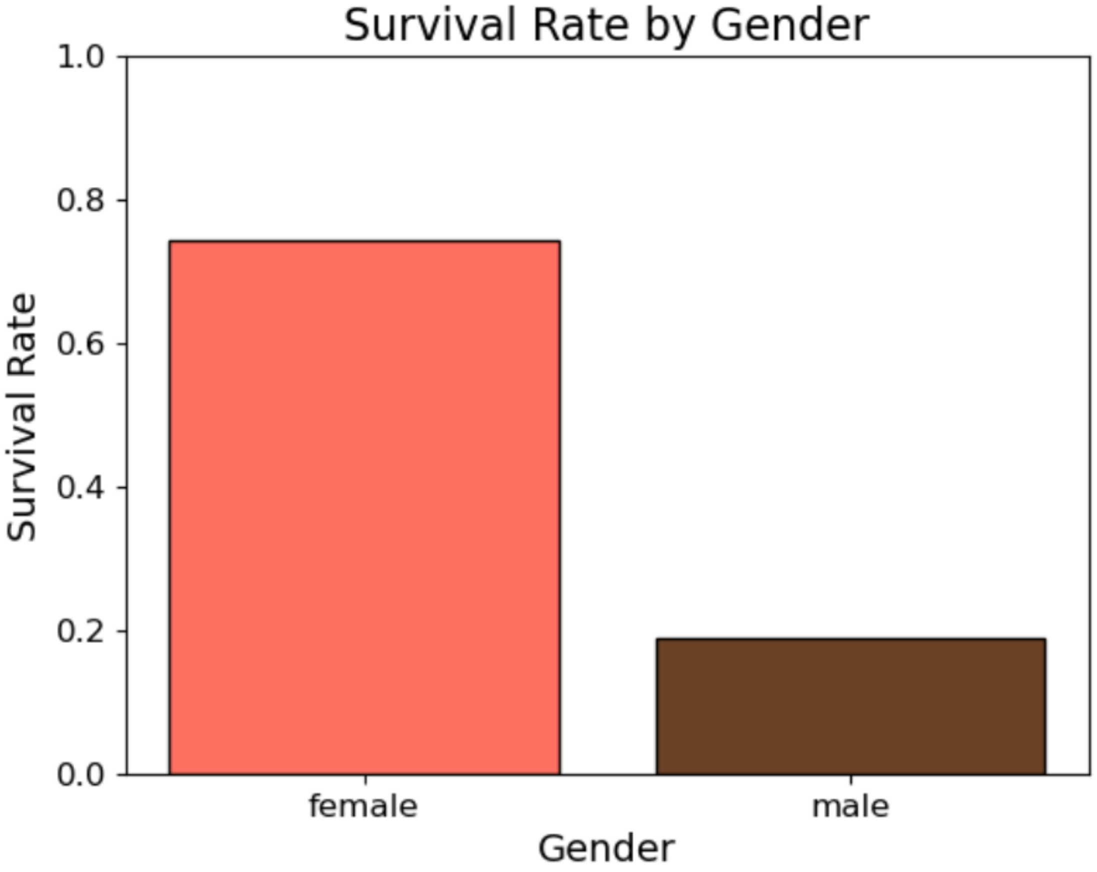
Titanic Dataset Analysis with Visualization

- **Exploring Data**



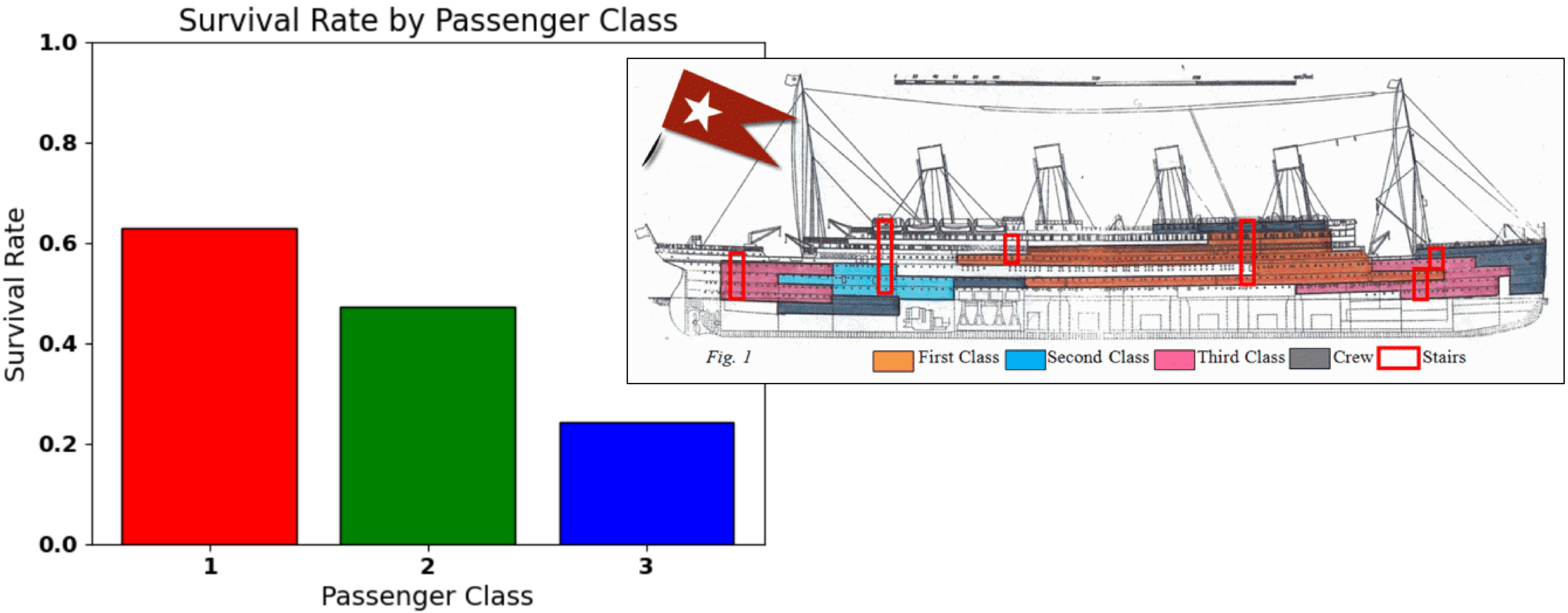
Titanic Dataset Analysis with Visualization

- **Did gender affect survival?**



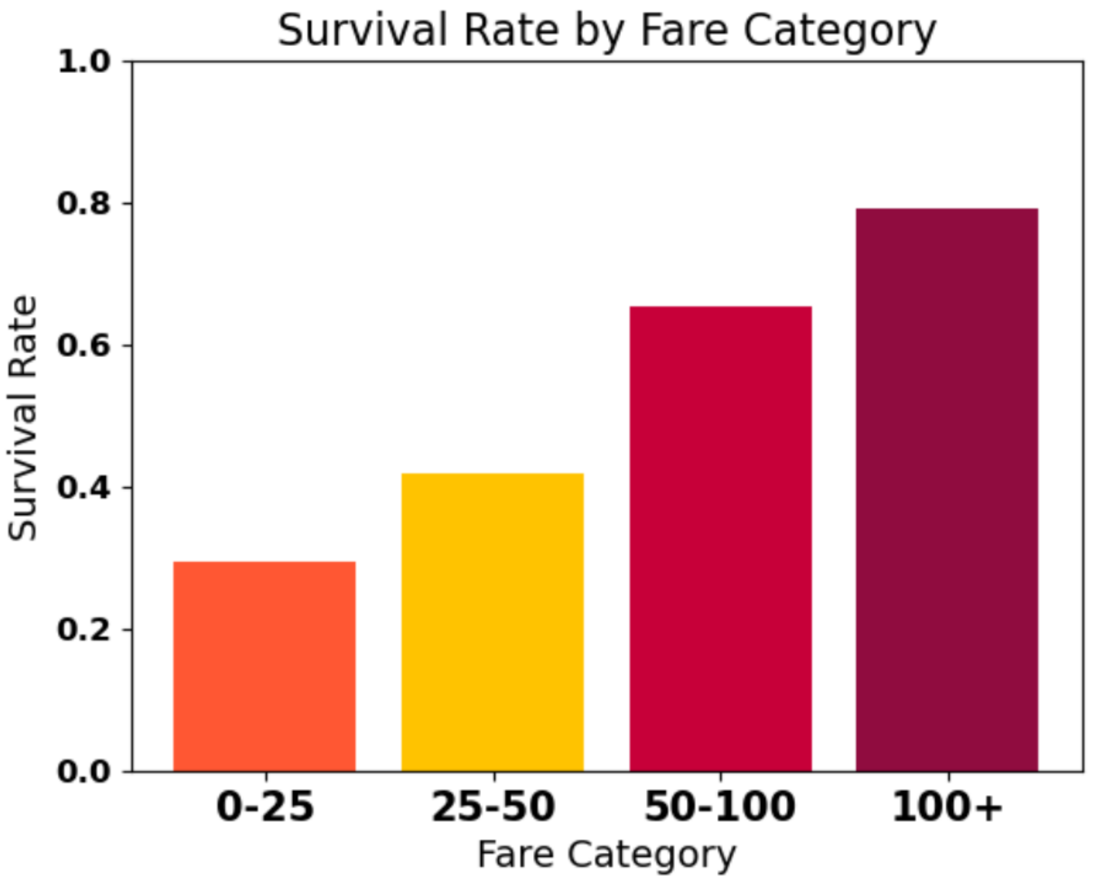
Titanic Dataset Analysis with Visualization

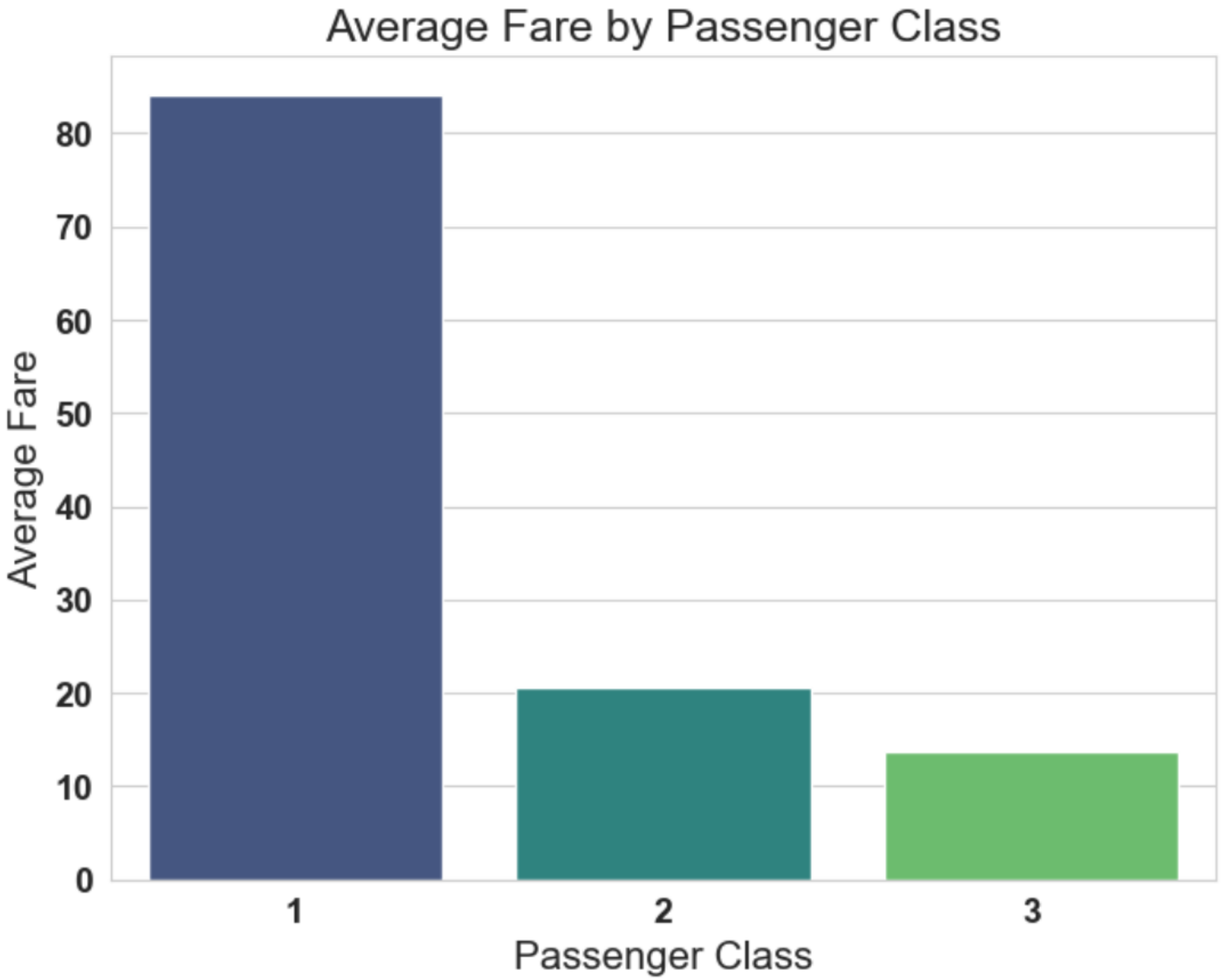
- Did passenger class affect survival?



Titanic Dataset Analysis with Visualization

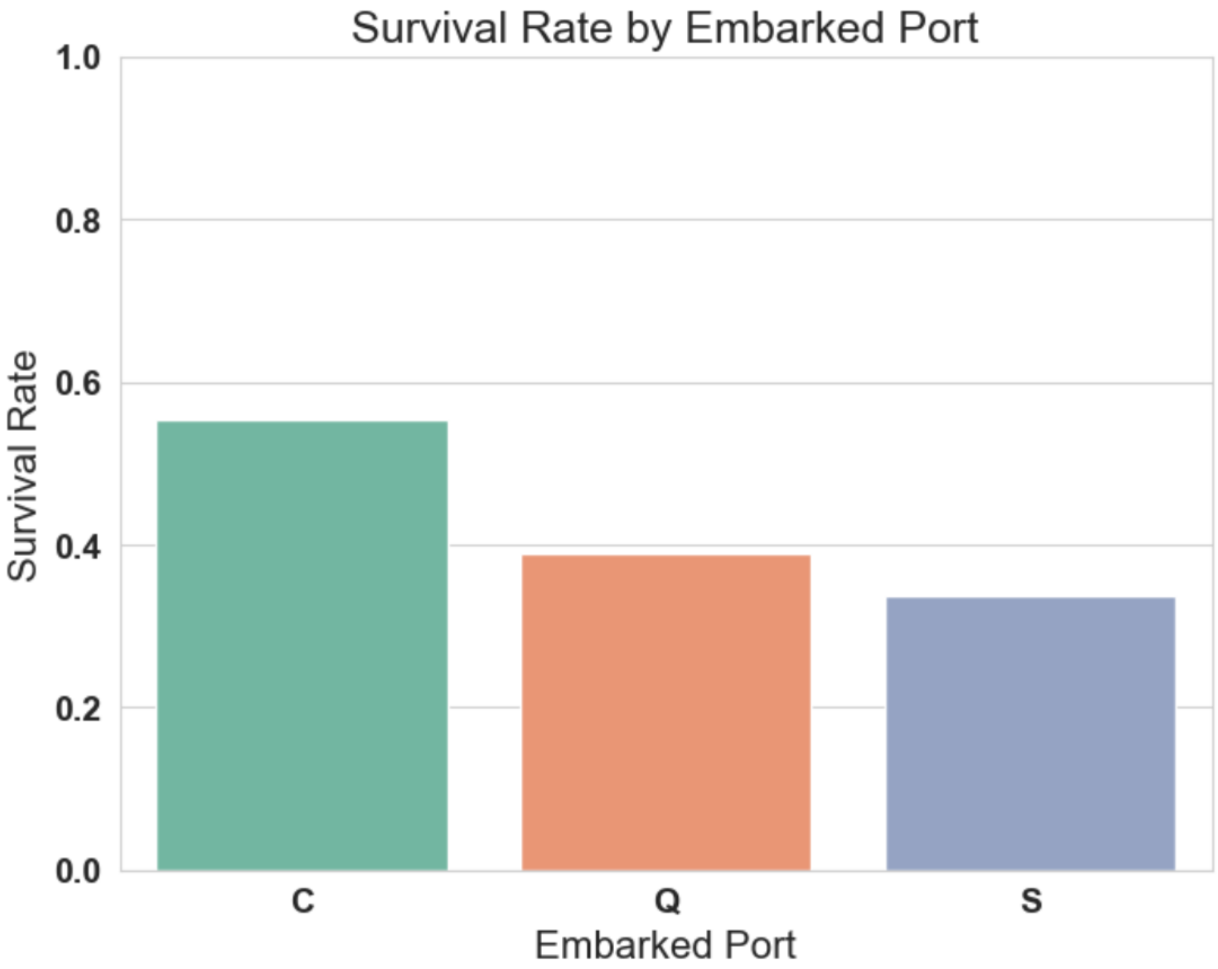
- Did fare affect survival?

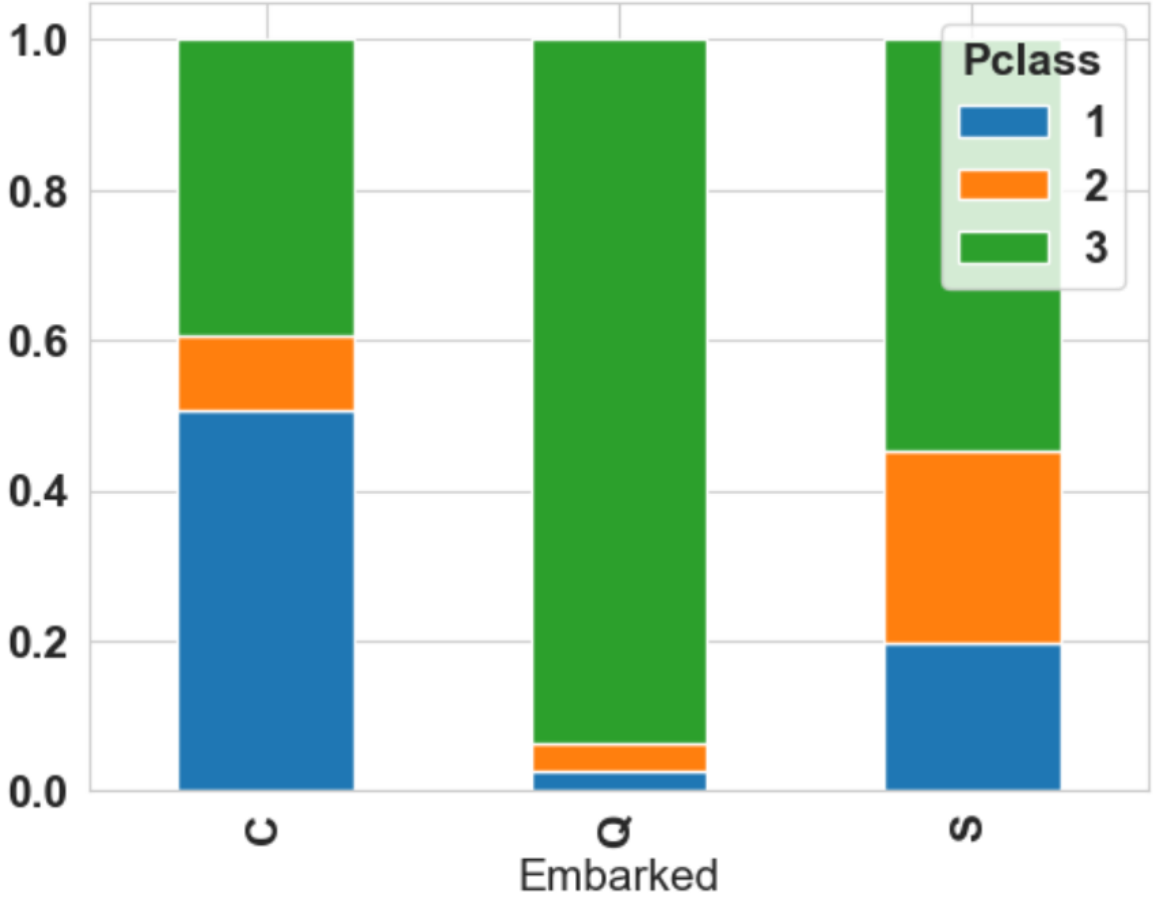




Titanic Dataset Analysis with Visualization

- Did embarked port affect survival?





Titanic Dataset Analysis with Visualization

- **What characteristics are likely to have impact on survival?**
  - **Gender (Sex)**
    - Female passenger) had a higher chance of survival compared to male passengers
  - **Passenger Class (Pclass)**
    - Passengers in higher classes were more likely to survive than those in lower classes
  - **Fare**
    - Passengers who pay the higher price are in a higher class. So it has an impact for the same reasons as above
  - **Embarked Port (Embarked)**
    - The port where a passenger boarded the Titanic (Southampton, Cherbourg, or Queenstown) might have some influence on survival, although this is not as strong a predictor as some other factors.

Python for Data Visualization

- matplotlib
  - Python 2D plotting library
    - line plots, scatter plots, barcharts, histograms, pie charts etc.
  - Producing publication quality figures in a variety of hardcopy formats
  - A set of functionalities similar to those of MATLAB

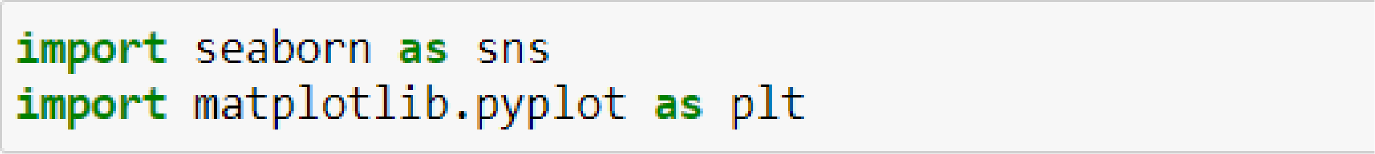
  - Relatively low-level; some effort needed to create advanced visualization

Python for Data Visualization

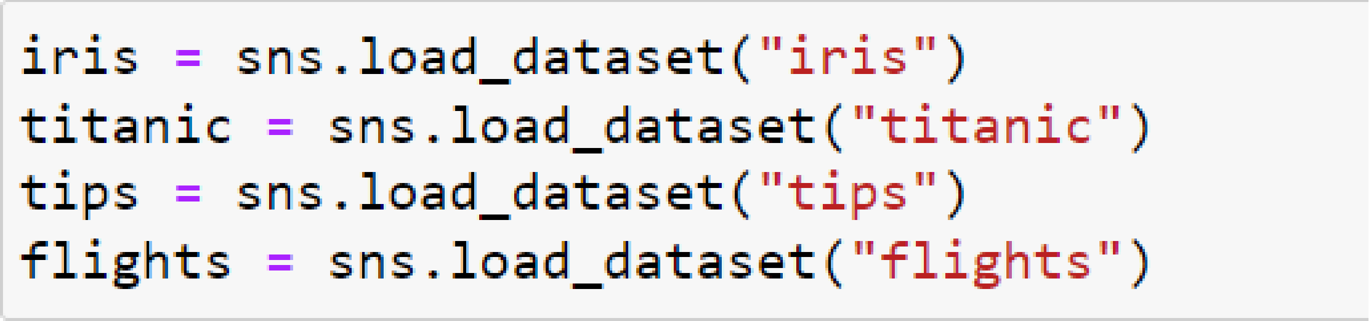
- seaborn
  - Python visualization library based on Matplotlib
  - Provides high level interface for drawing attractive *statistical graphics*
  - Similar (in style) to the popular ggplot2 library in R

Seaborn

- **load_dataset()**
  - Load an example dataset from the online repository (requires internet)



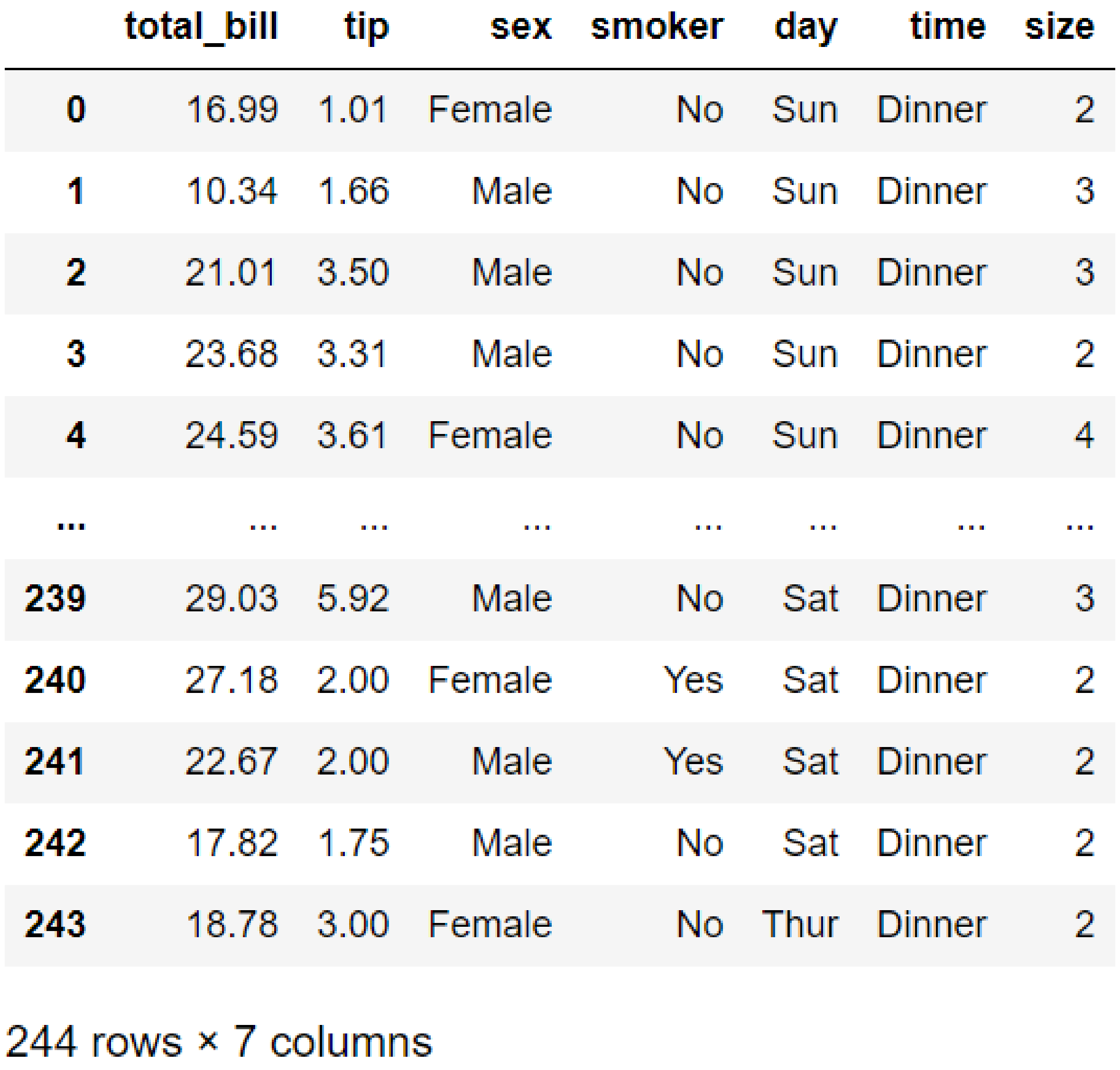
```python
import seaborn as sns
import matplotlib.pyplot as plt
```



```python
iris = sns.load_dataset("iris")
titanic = sns.load_dataset("titanic")
tips = sns.load_dataset("tips")
flights = sns.load_dataset("flights")
```

Seaborn

- **load_dataset()**
  - Tips dataset

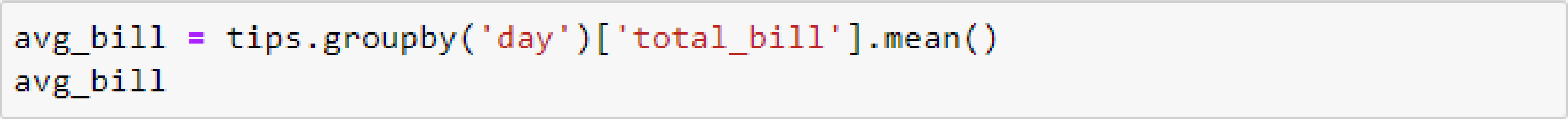


	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

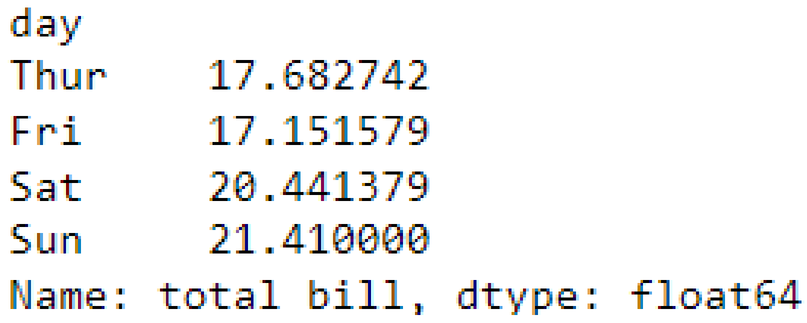
244 rows × 7 columns

Seaborn

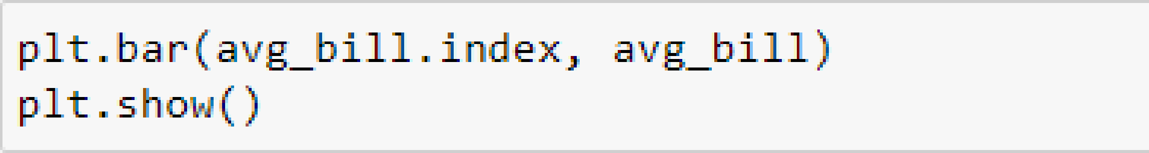
- **Bar Plot : Average total bill per Day**



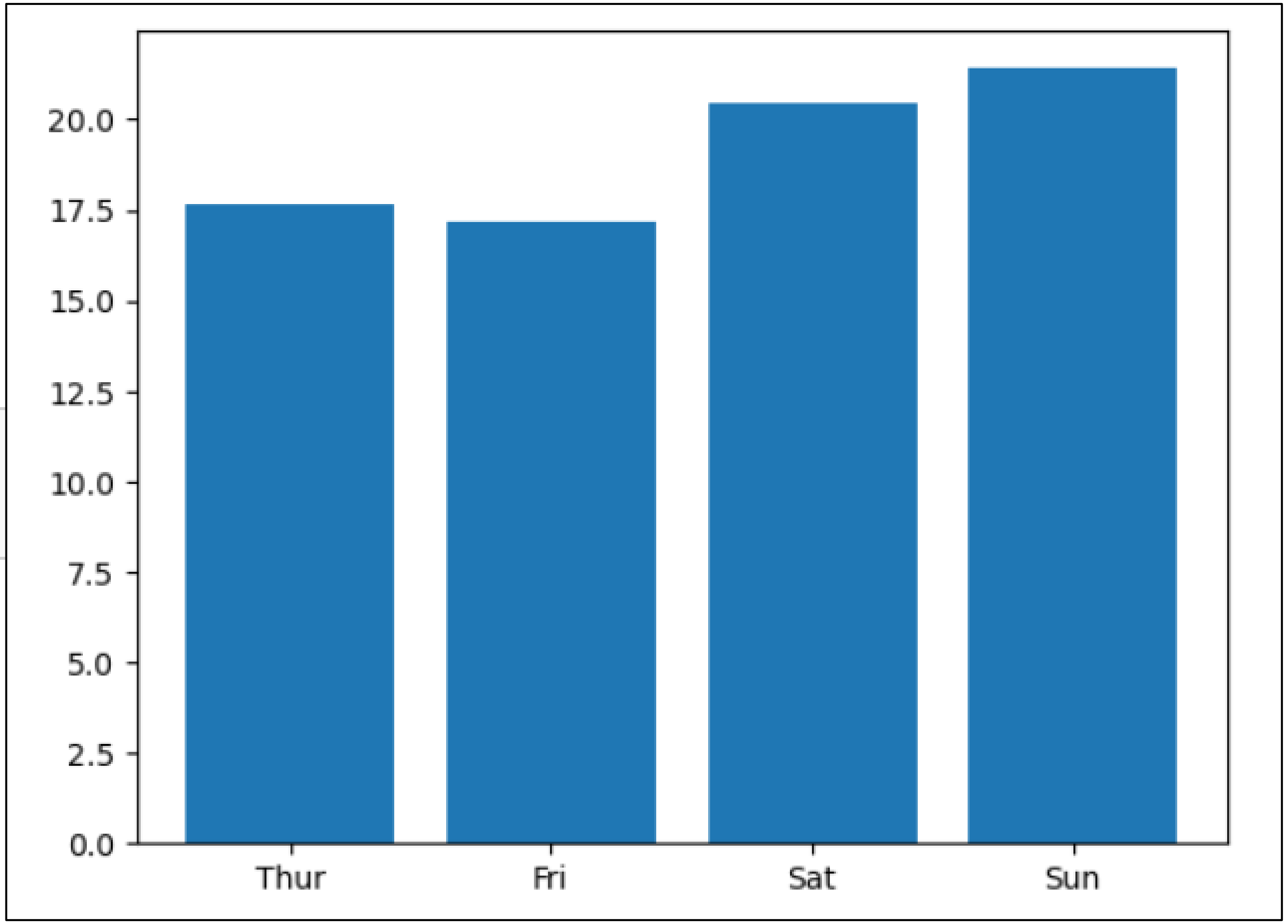
```
avg_bill = tips.groupby('day')['total_bill'].mean()
avg_bill
```



```
day
Thur    17.682742
Fri     17.151579
Sat     20.441379
Sun     21.410000
Name: total bill, dtype: float64
```

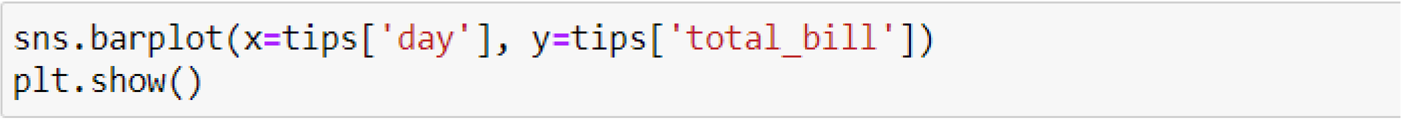


```
plt.bar(avg_bill.index, avg_bill)
plt.show()
```

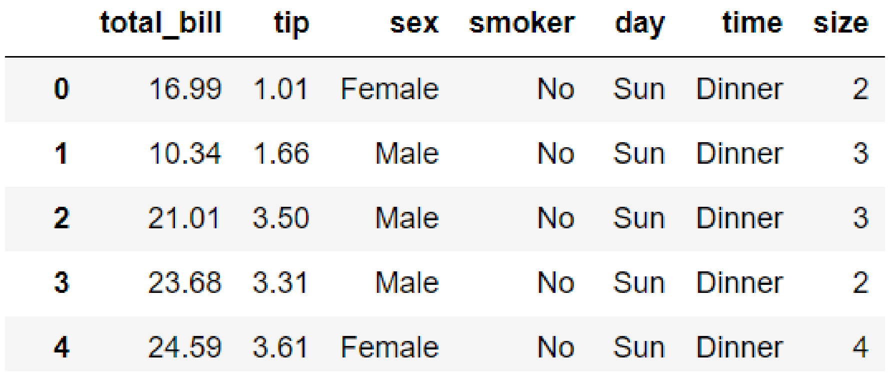


Seaborn

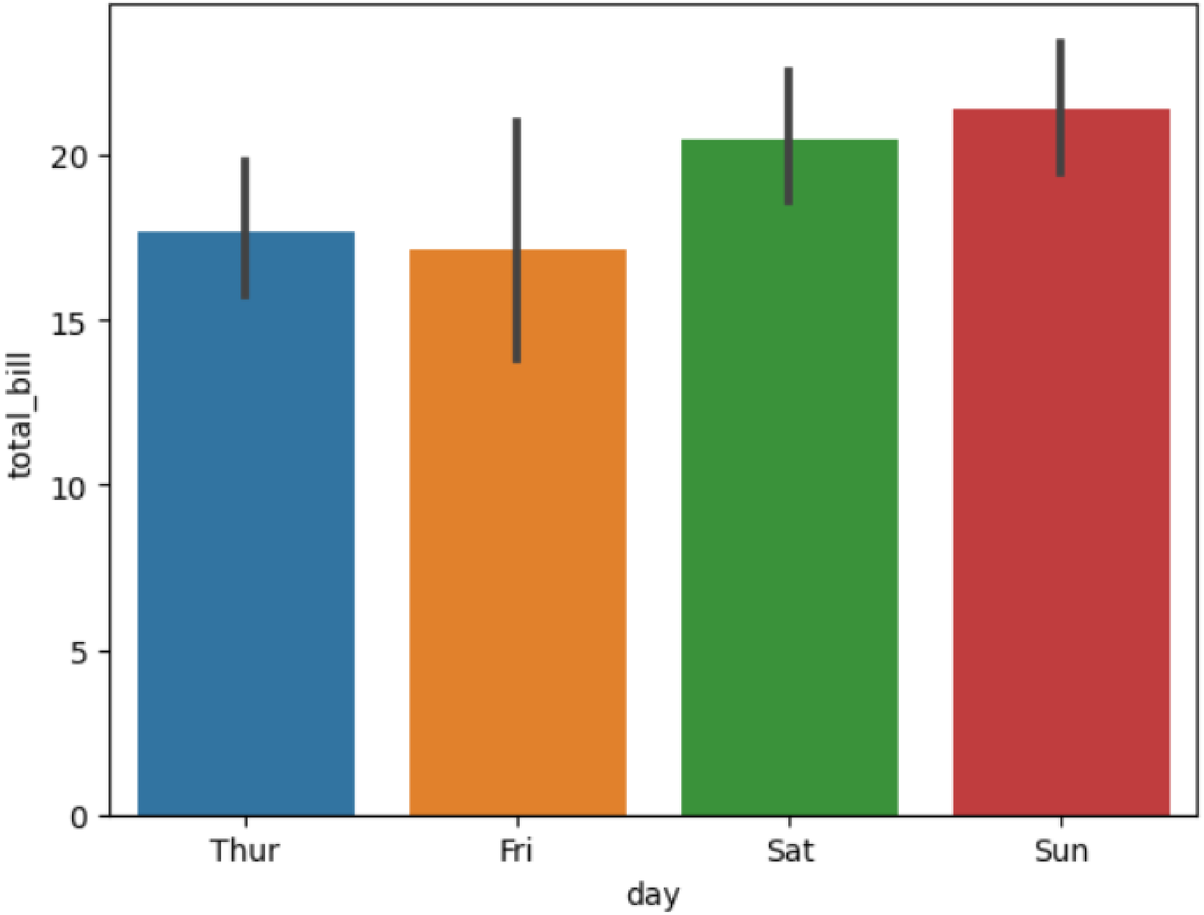
- **Bar Plot : Average total bill per Day**

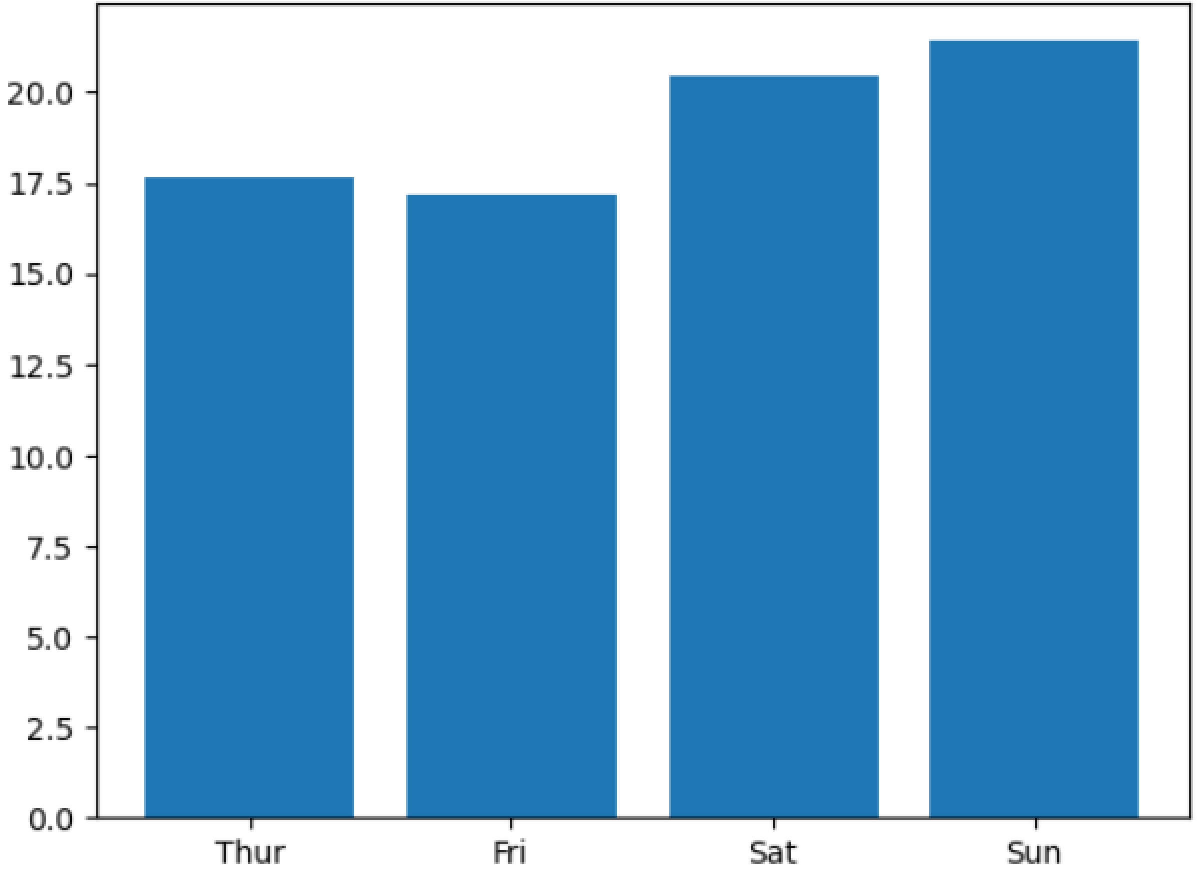


```
sns.barplot(x=tips['day'], y=tips['total_bill'])
plt.show()
```



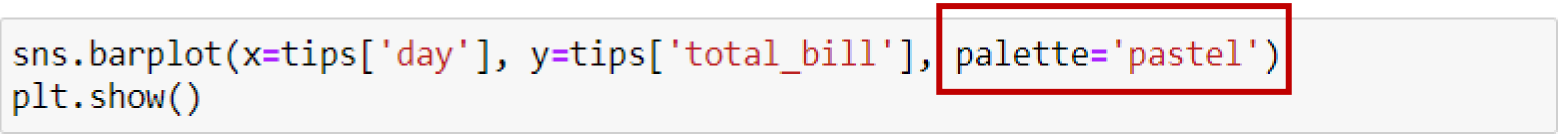
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4



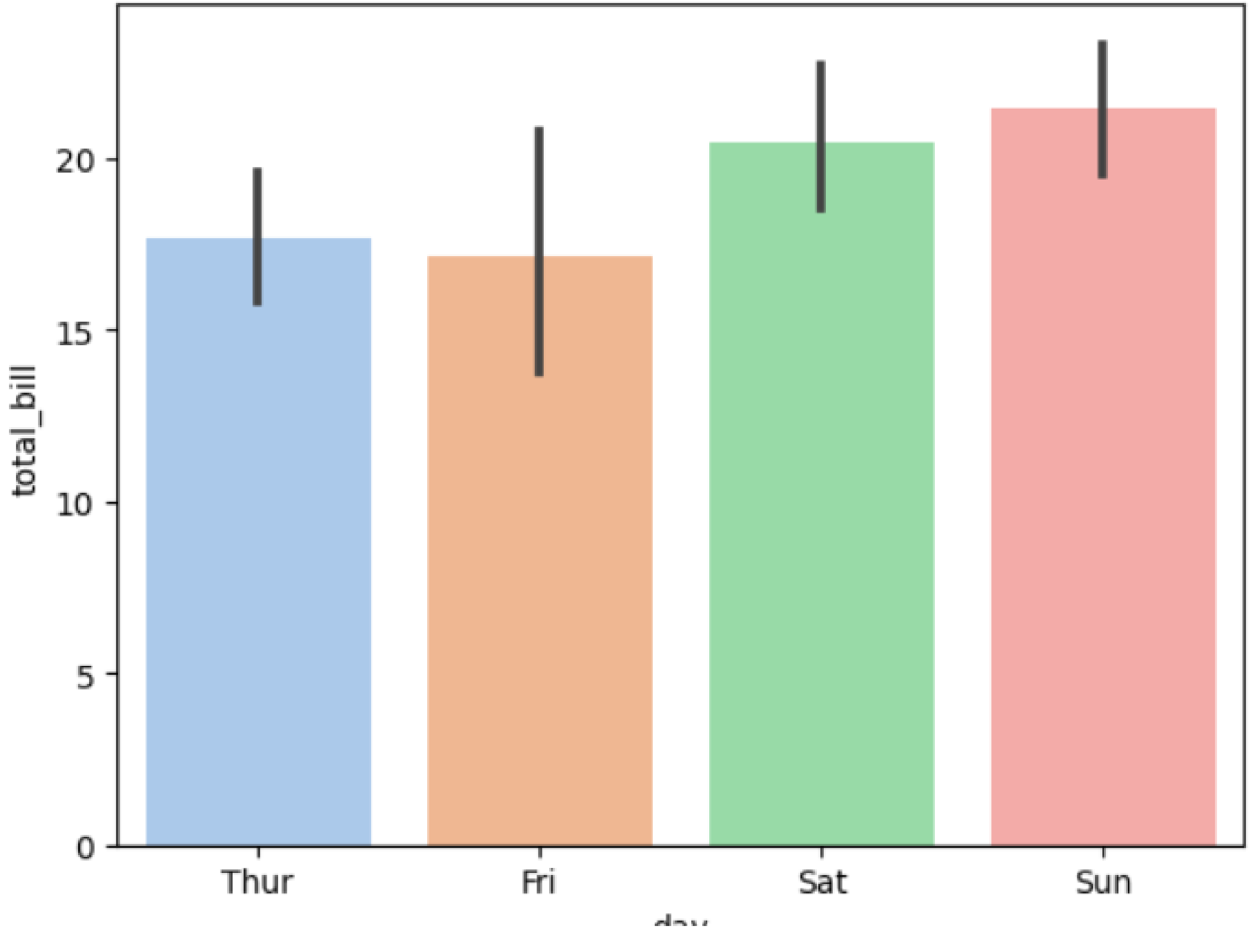


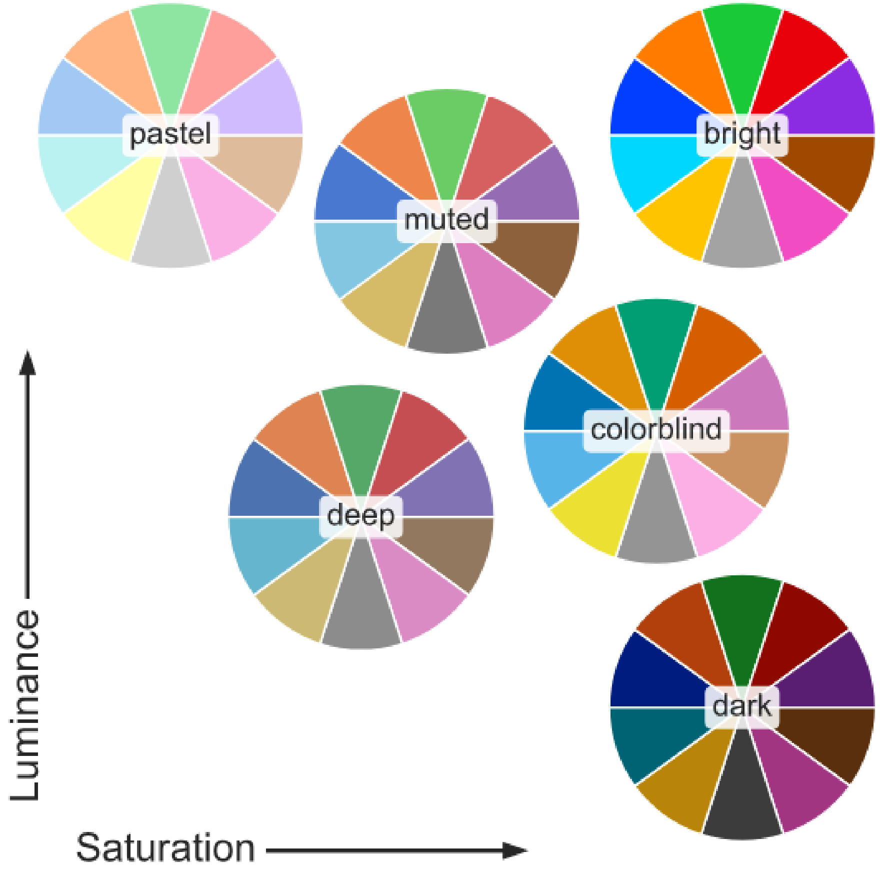
Seaborn

- ## Bar Plot



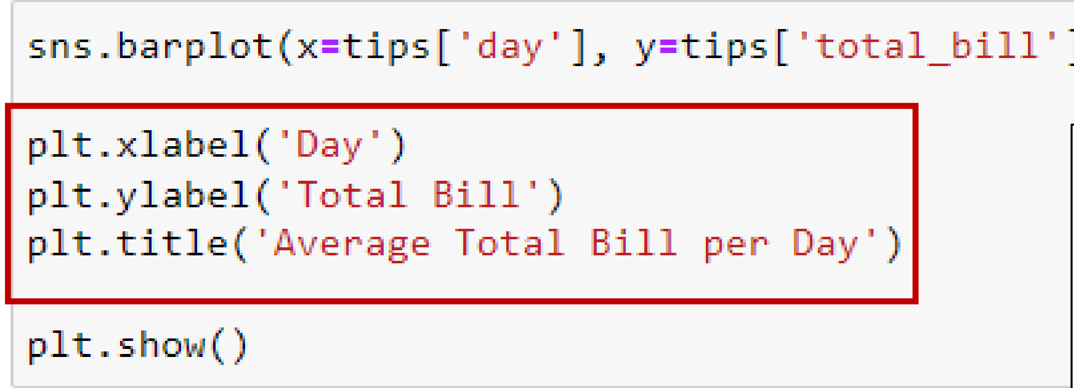
```
sns.barplot(x=tips['day'], y=tips['total_bill'], palette='pastel')
plt.show()
```



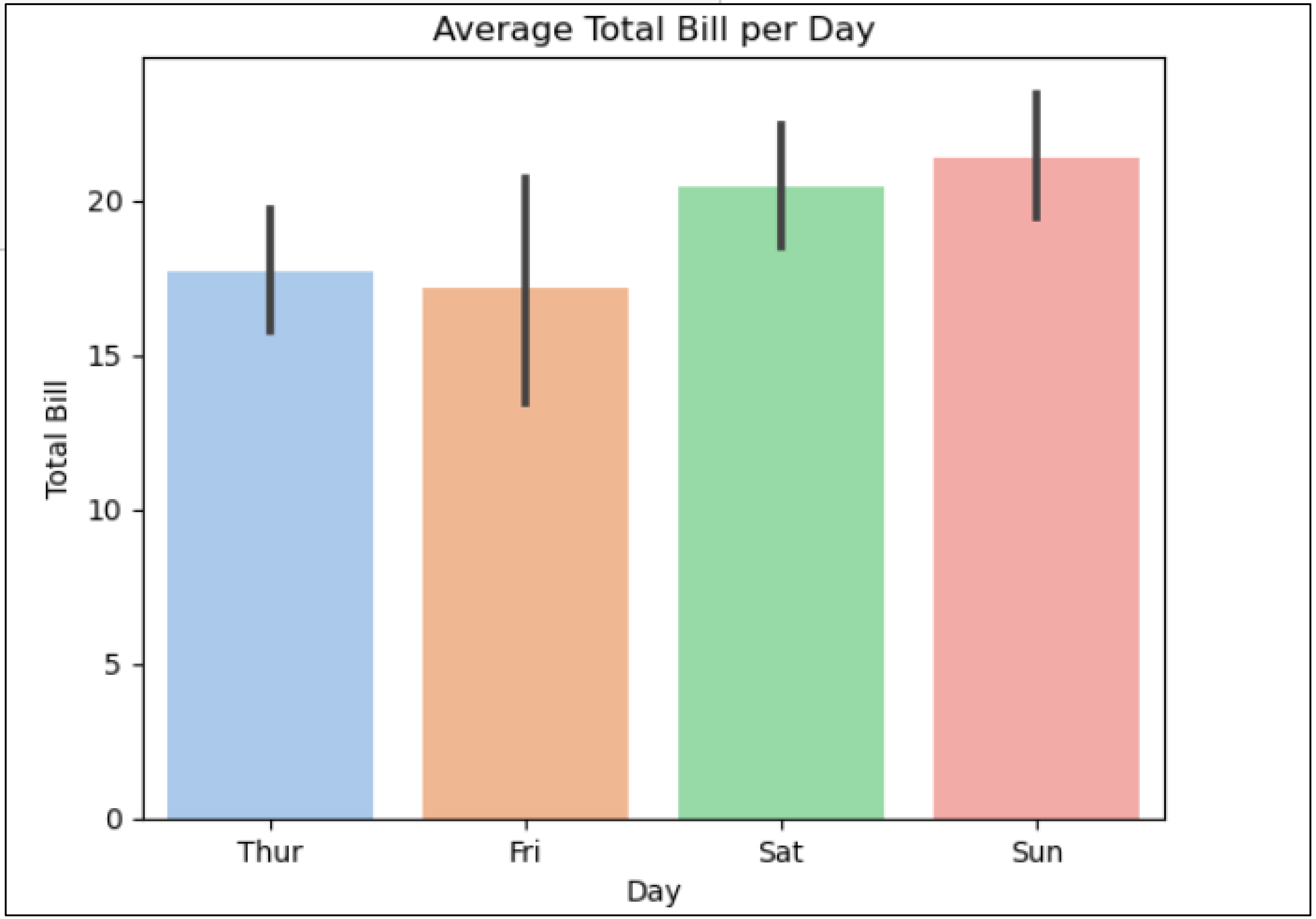


Seaborn

- **Bar Plot**

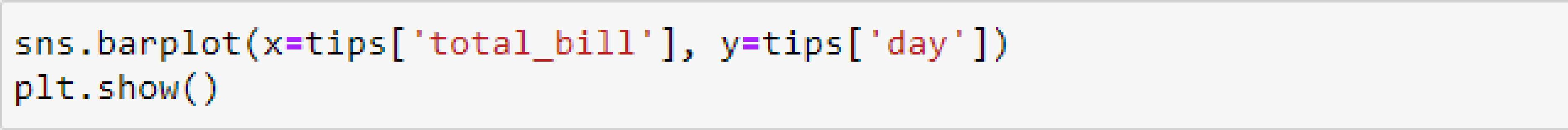


```
sns.barplot(x=tips['day'], y=tips['total_bill'], palette='pastel')

plt.xlabel('Day')
plt.ylabel('Total Bill')
plt.title('Average Total Bill per Day')

plt.show()
```

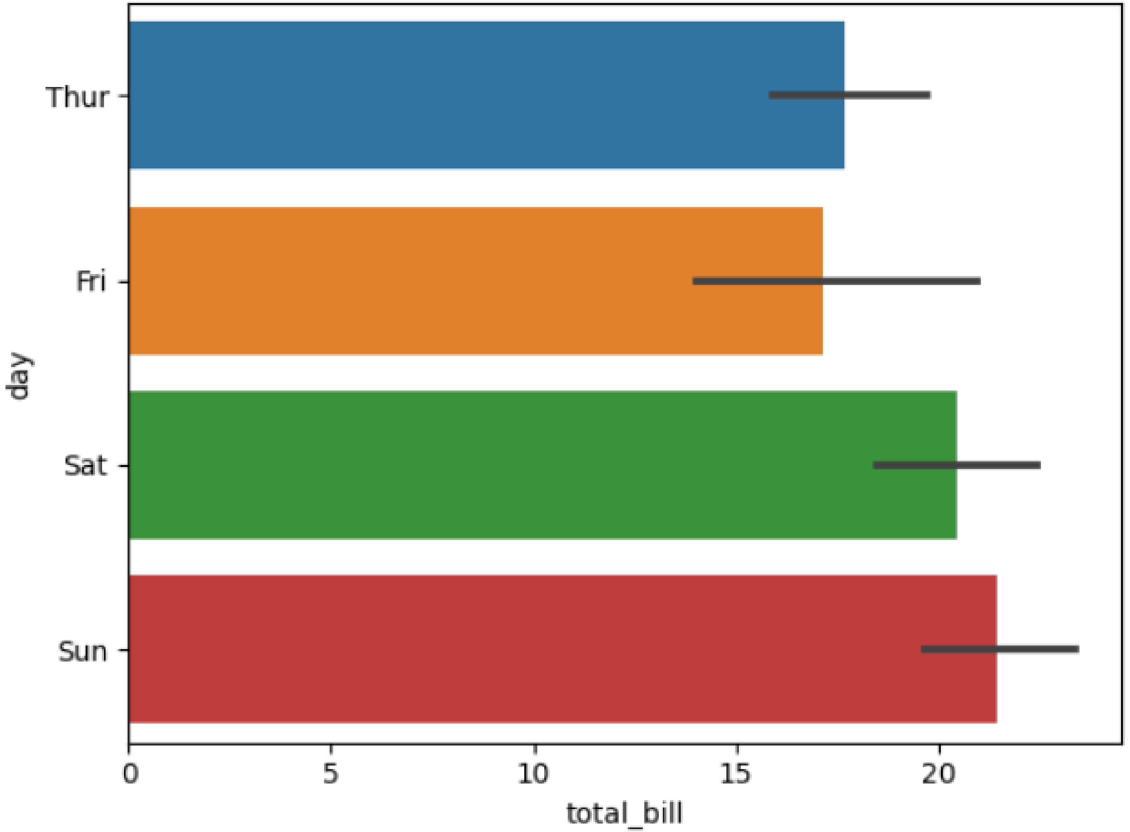


Seaborn

- **Bar Plot**

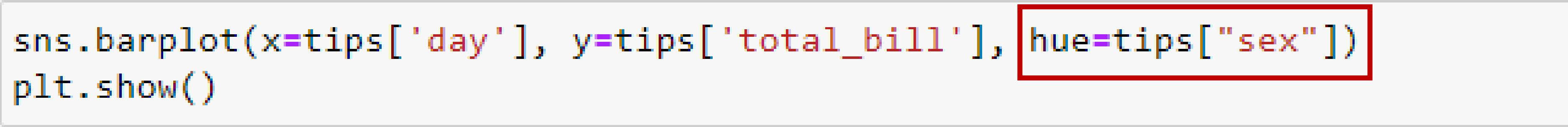


```
sns.barplot(x=tips['total_bill'], y=tips['day'])
plt.show()
```

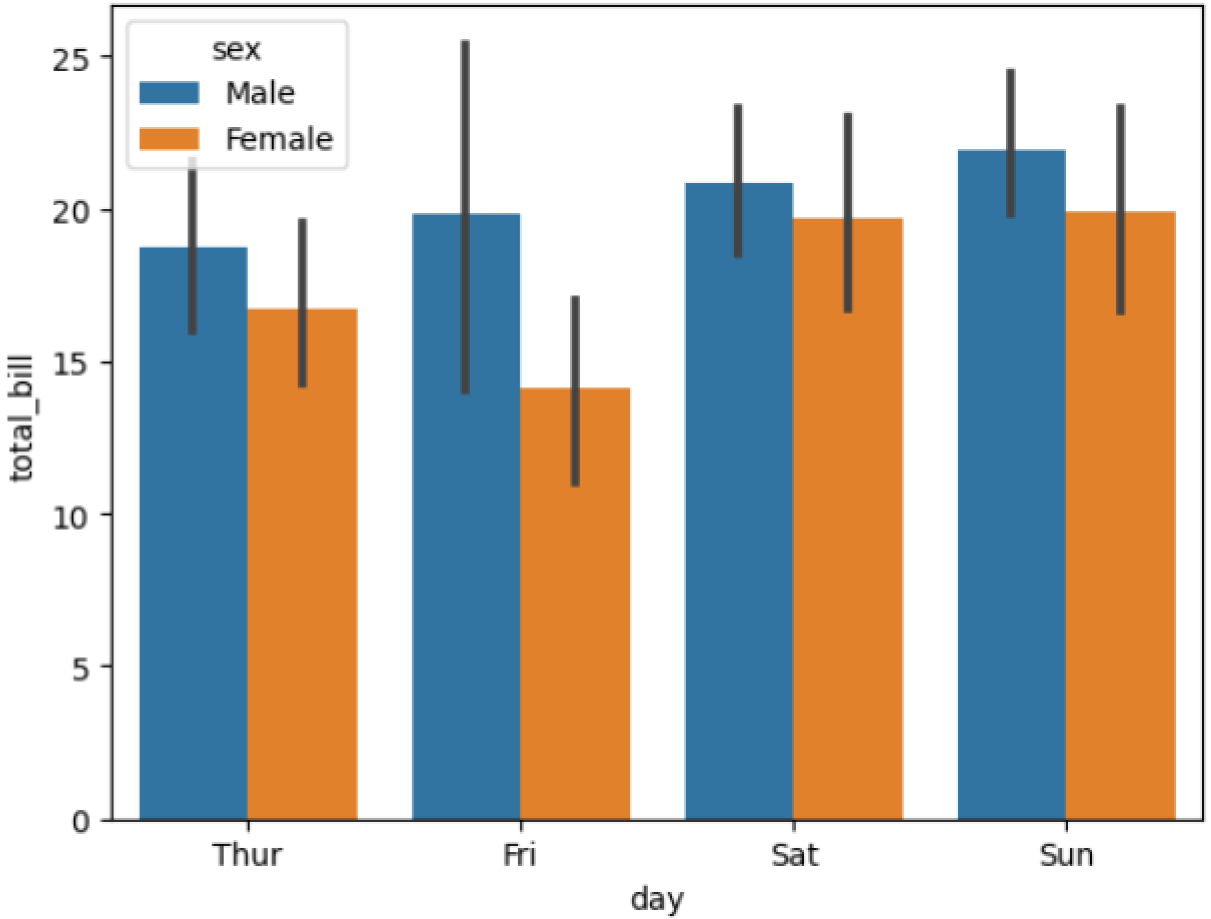


Seaborn

- ## Bar Plot

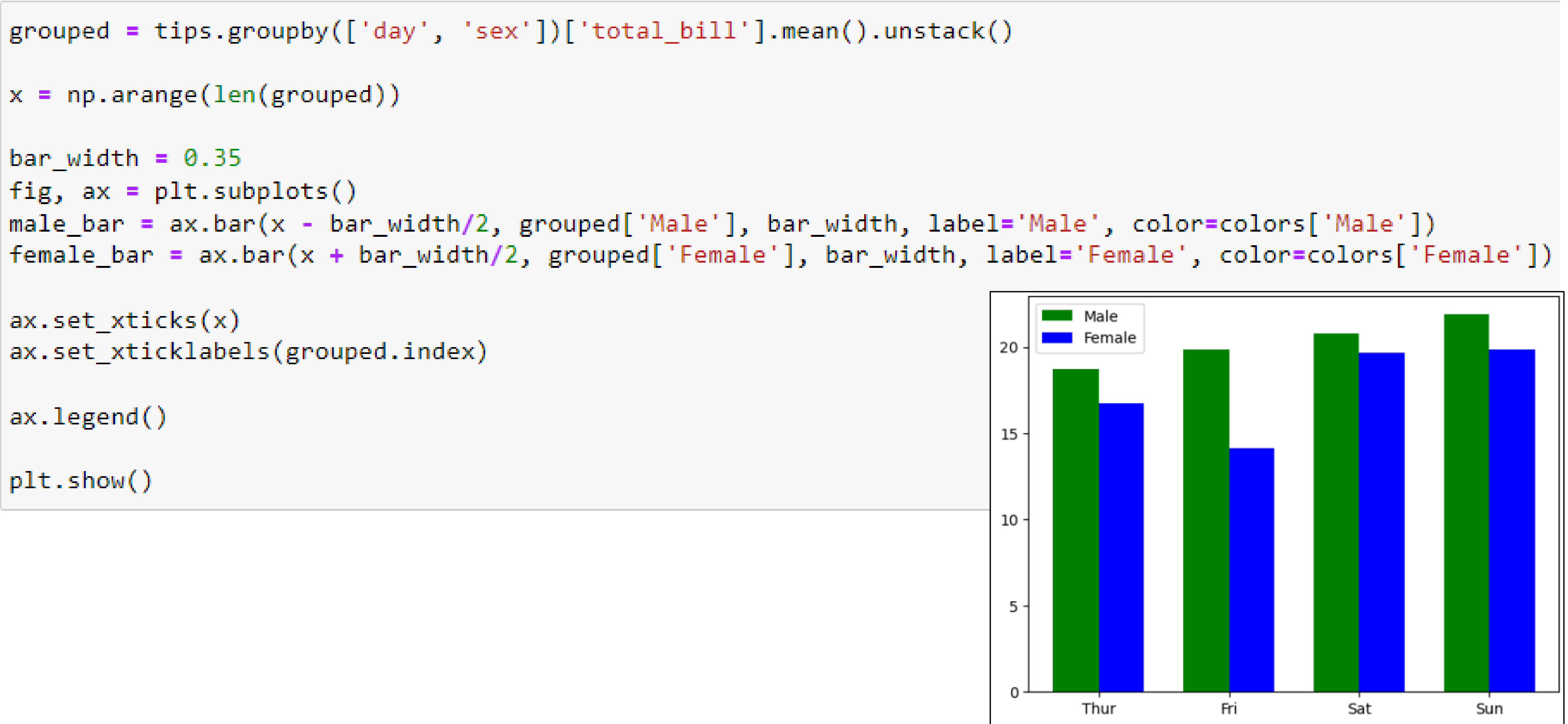


```
sns.barplot(x=tips['day'], y=tips['total_bill'], hue=tips["sex"])
plt.show()
```



Seaborn

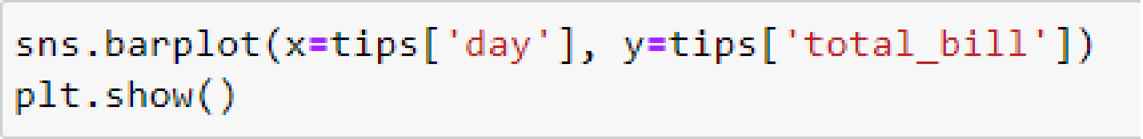
- ## Bar Plot



```
grouped = tips.groupby(['day', 'sex'])['total_bill'].mean().unstack()

x = np.arange(len(grouped))

bar_width = 0.35
fig, ax = plt.subplots()
male_bar = ax.bar(x - bar_width/2, grouped['Male'], bar_width, label='Male', color=colors['Male'])
female_bar = ax.bar(x + bar_width/2, grouped['Female'], bar_width, label='Female', color=colors['Female'])

ax.set_xticks(x)
ax.set_xticklabels(grouped.index)

ax.legend()

plt.show()
```

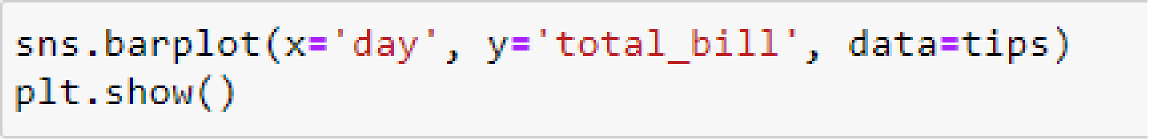
Seaborn

- **Bar Plot**

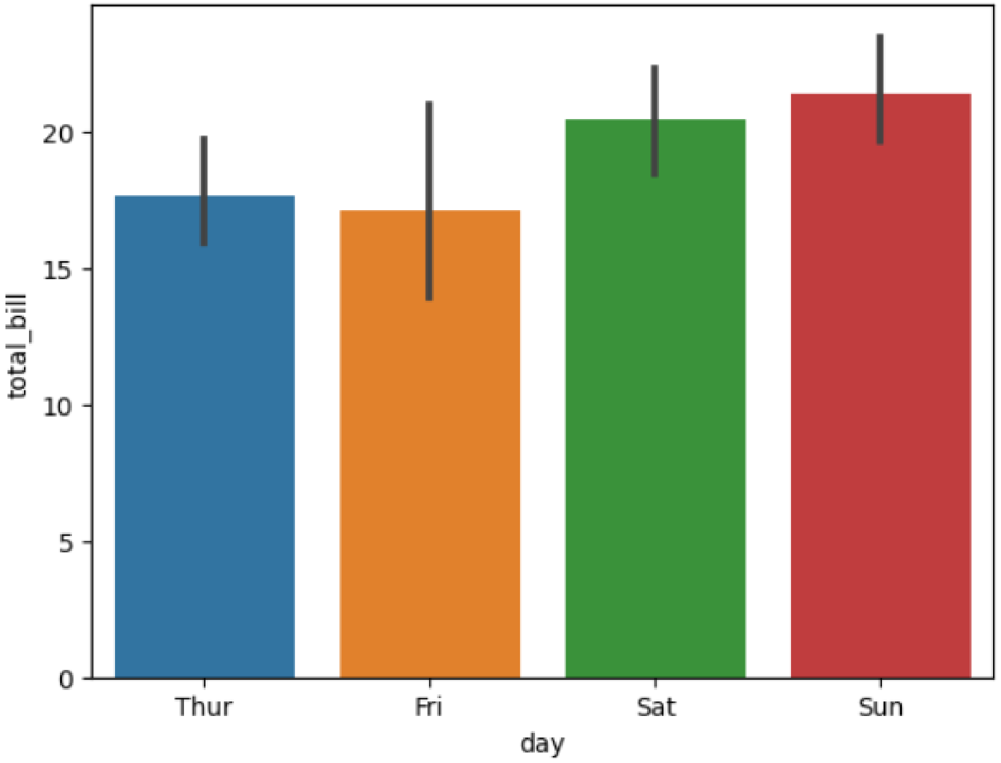


```
sns.barplot(x=tips['day'], y=tips['total_bill'])
plt.show()
```

=

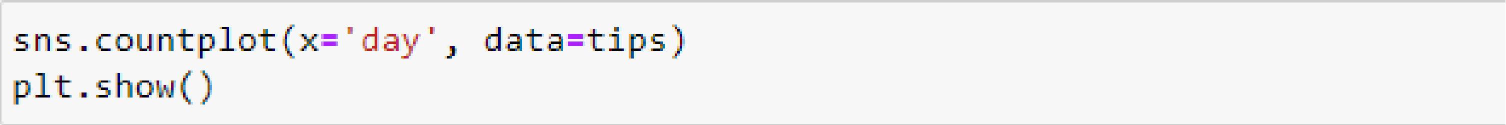


```
sns.barplot(x='day', y='total_bill', data=tips)
plt.show()
```

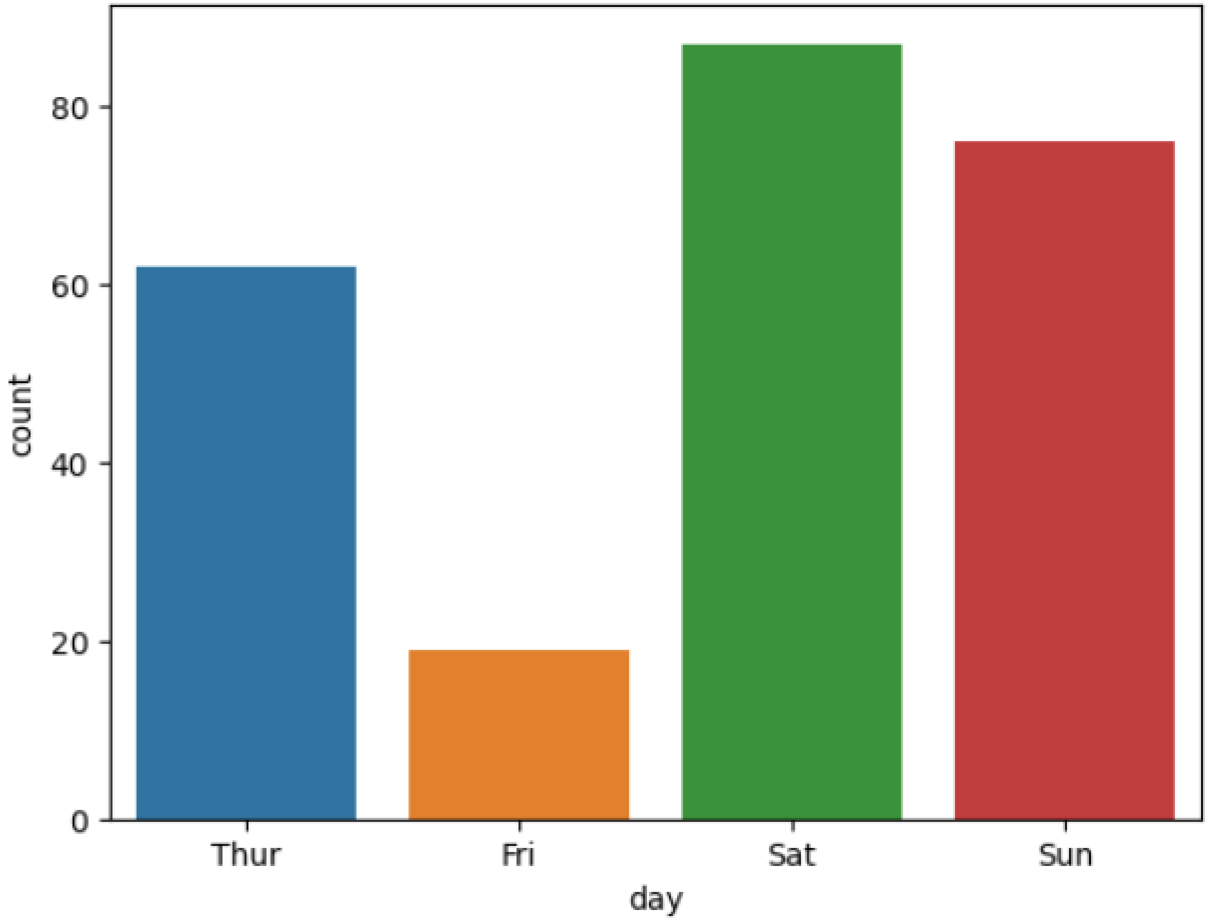


Seaborn

- Count Plot

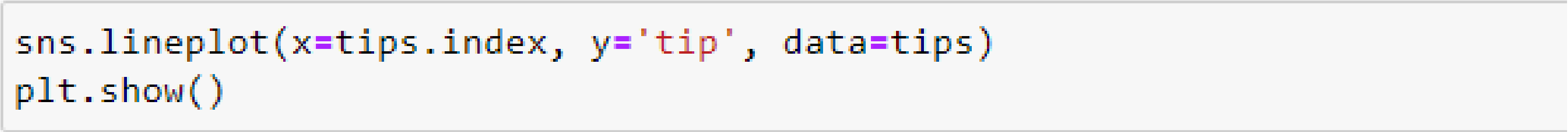


```
sns.countplot(x='day', data=tips)
plt.show()
```

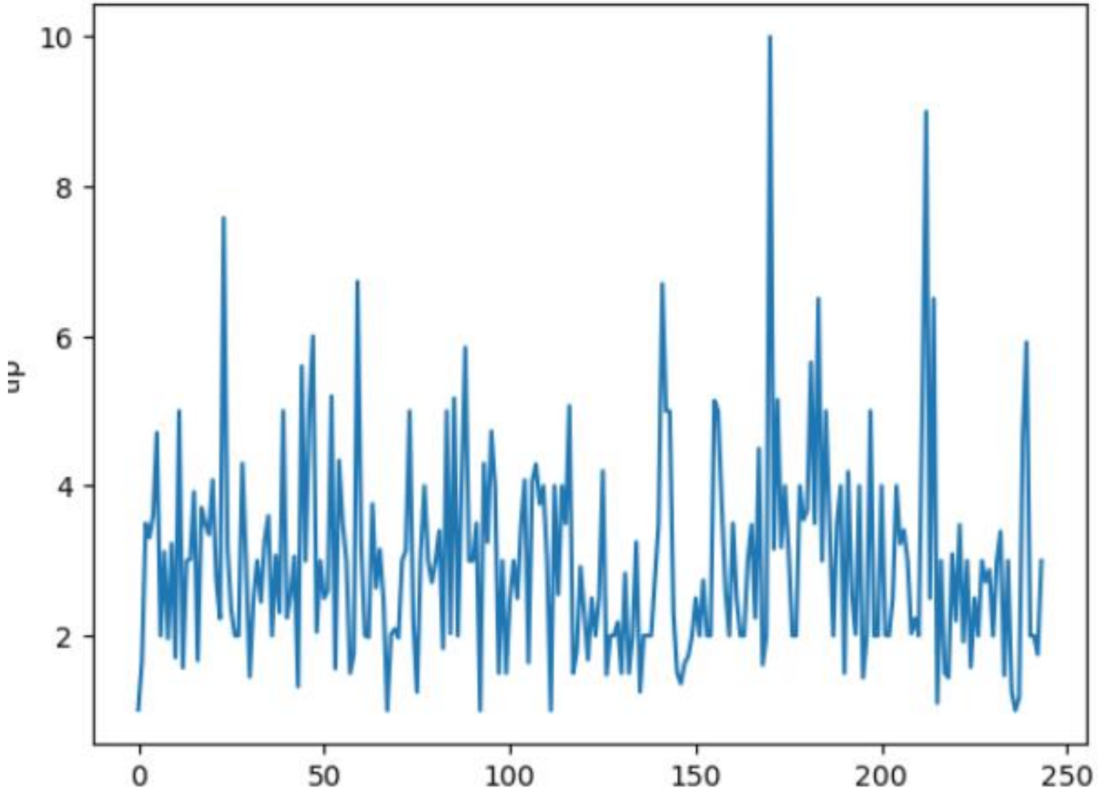


Seaborn

- **Line Plot**

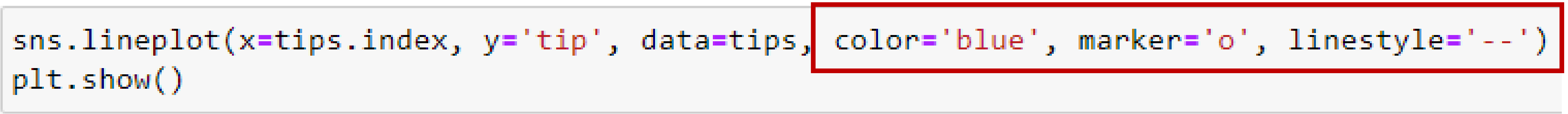


```
sns.lineplot(x=tips.index, y='tip', data=tips)
plt.show()
```

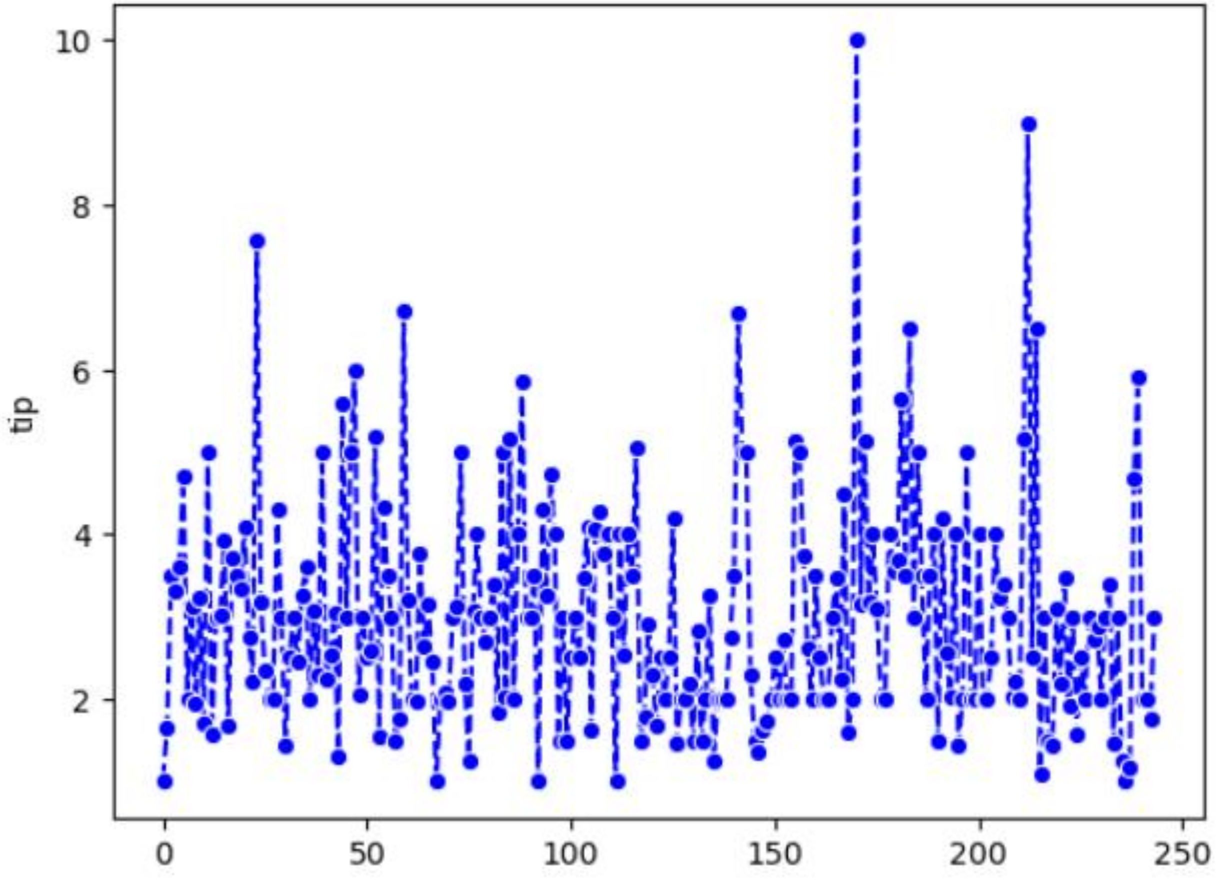


Seaborn

- **Line Plot**

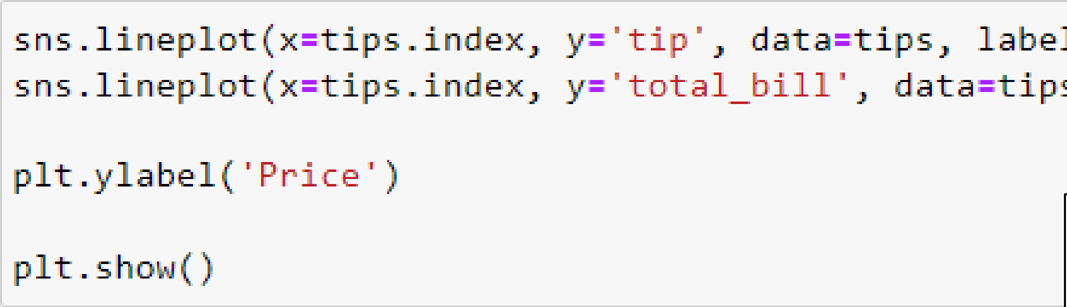


```
sns.lineplot(x=tips.index, y='tip', data=tips, color='blue', marker='o', linestyle='--')
plt.show()
```

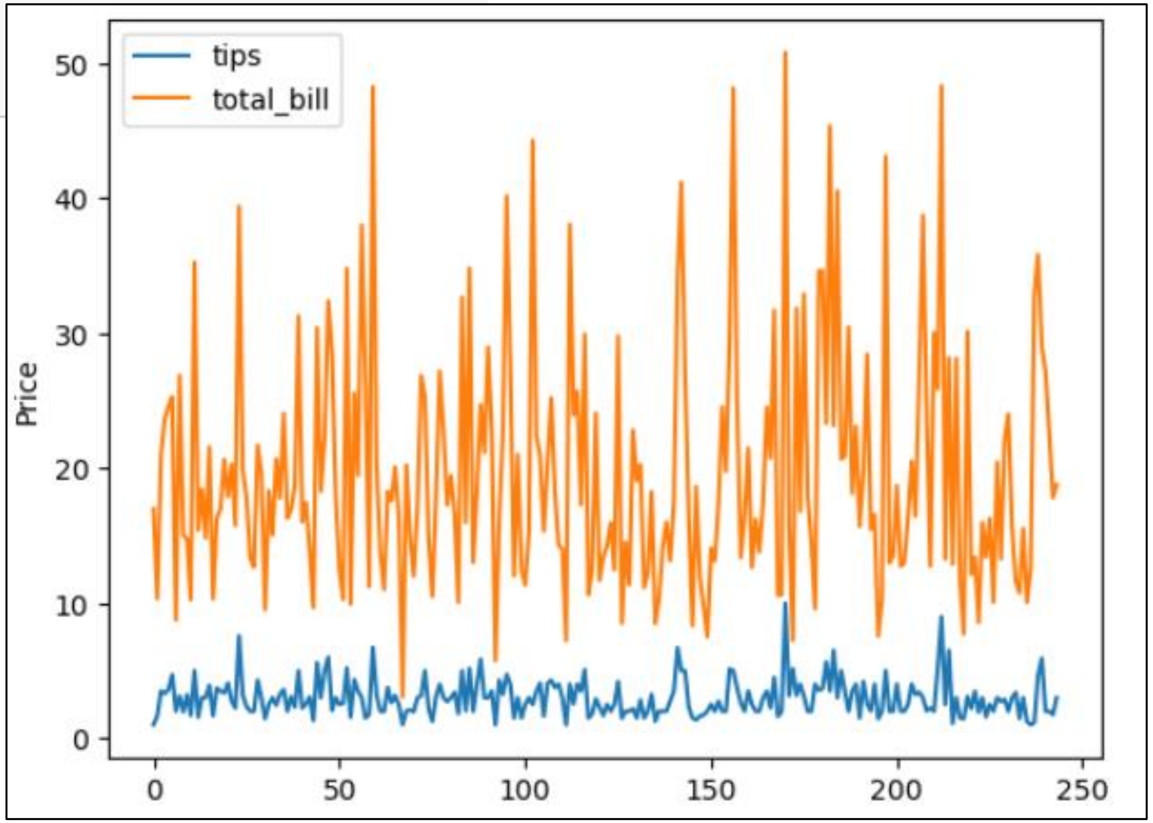


Seaborn

- **Line Plot**

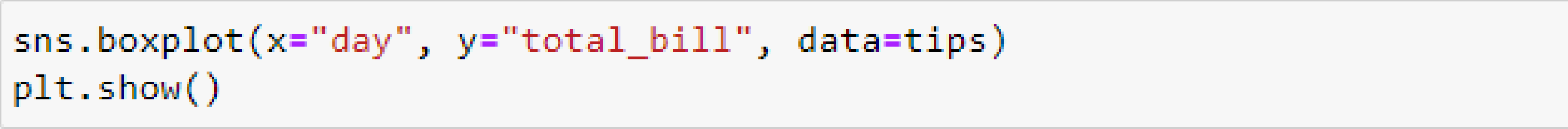


```
sns.lineplot(x=tips.index, y='tip', data=tips, label='tips')
sns.lineplot(x=tips.index, y='total_bill', data=tips, label='total_bill')

plt.ylabel('Price')

plt.show()
```

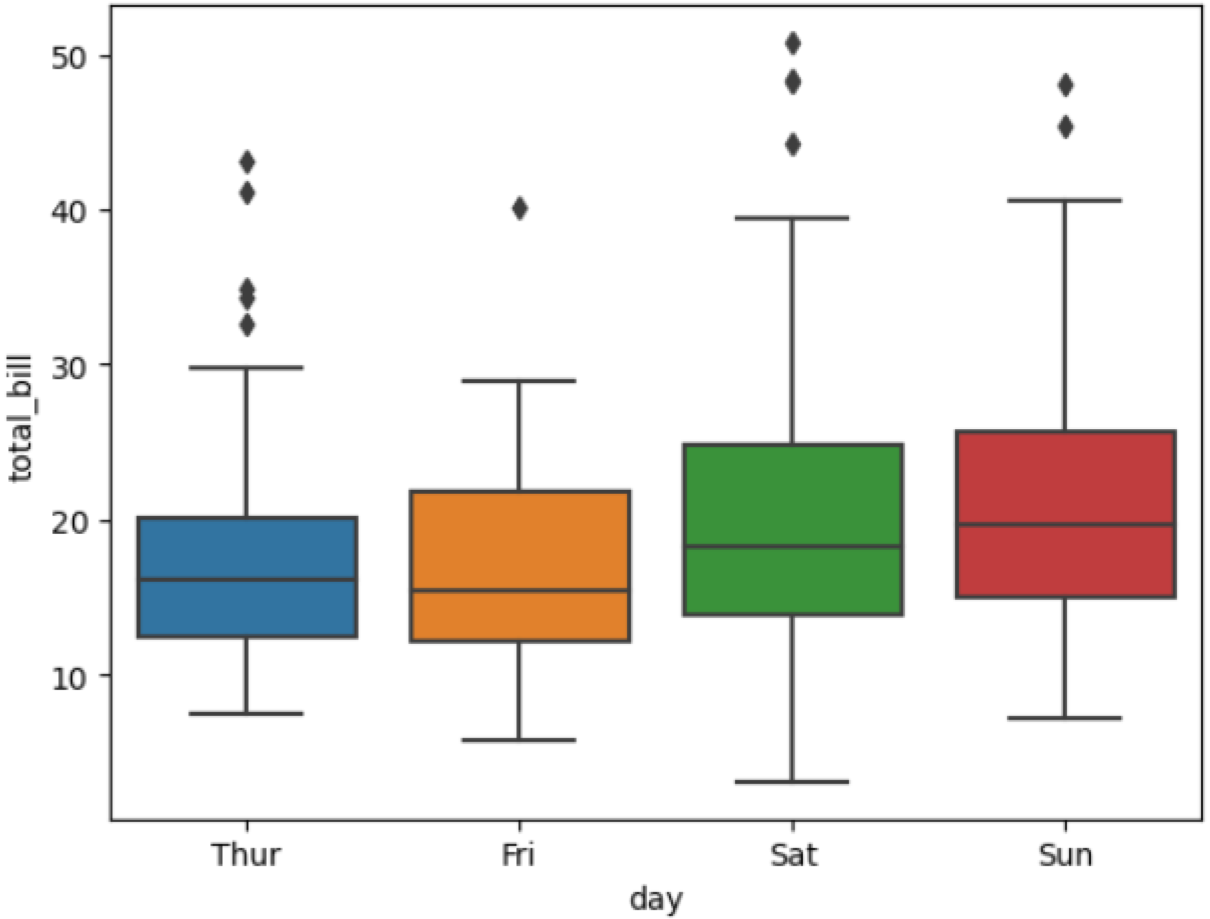


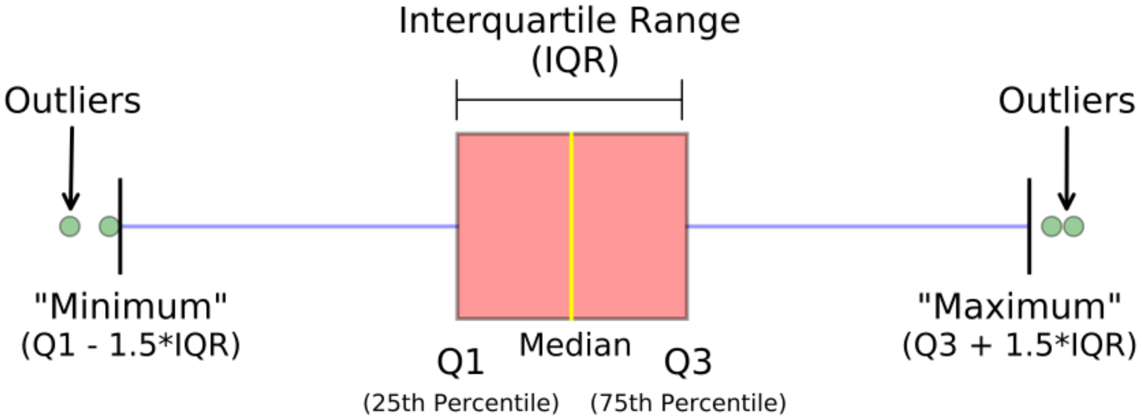
Seaborn

- **Box Plot**



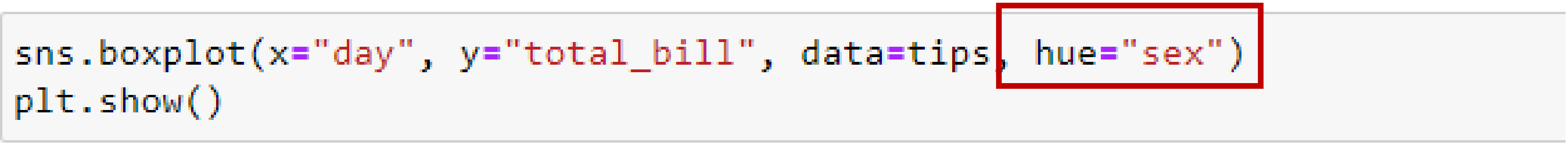
```
sns.boxplot(x="day", y="total_bill", data=tips)
plt.show()
```



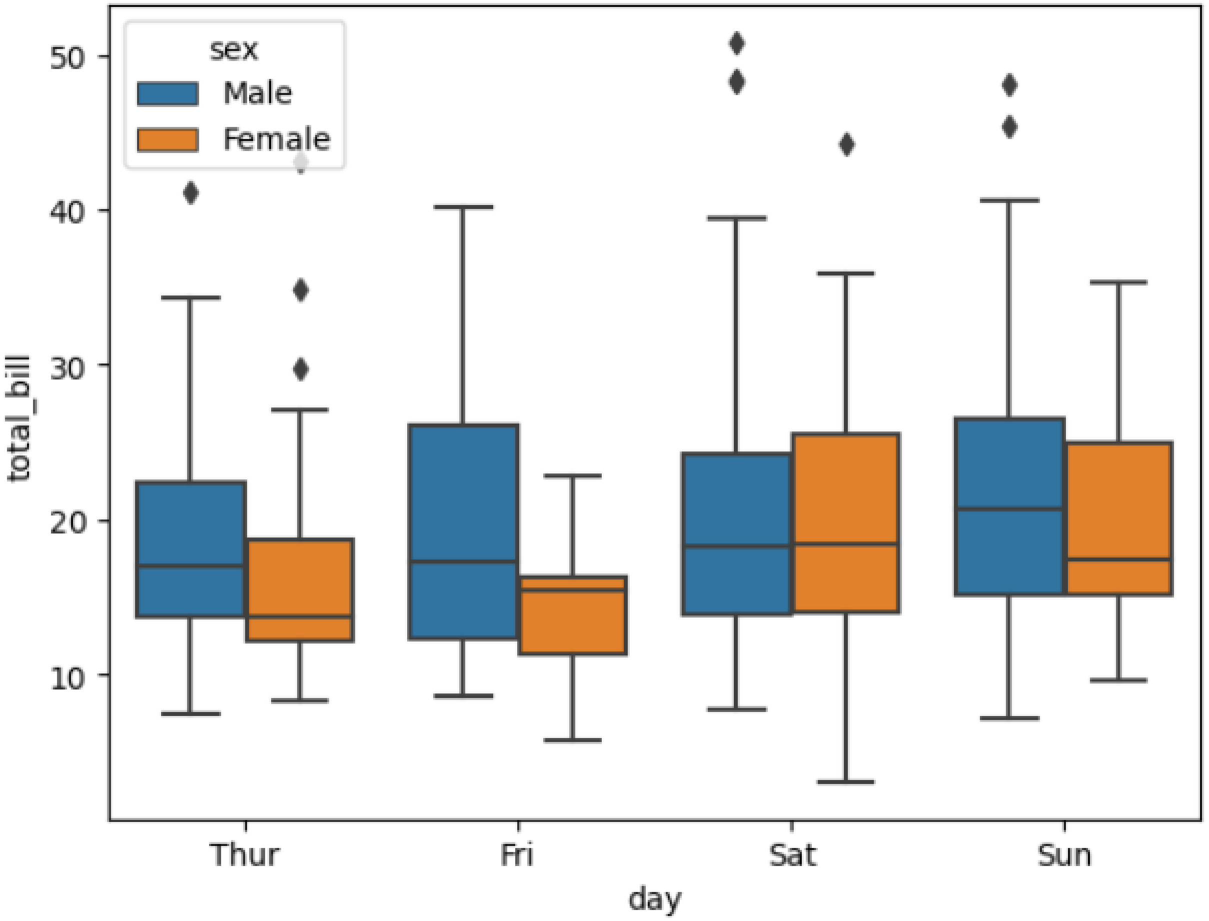


Seaborn

- **Box Plot**

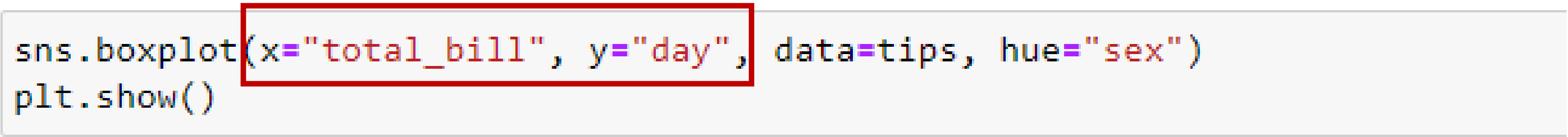


```
sns.boxplot(x="day", y="total_bill", data=tips, hue="sex")
plt.show()
```

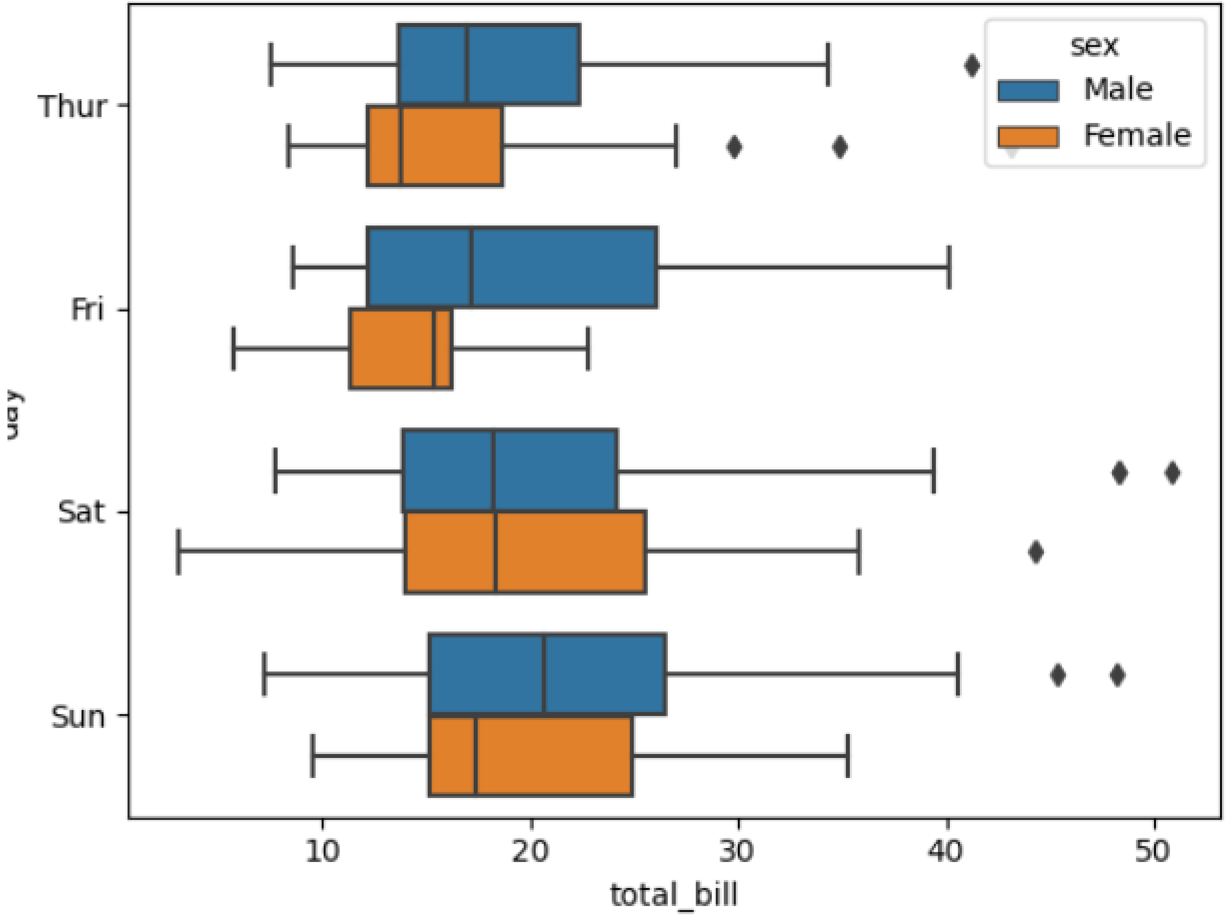


Seaborn

- **Box Plot**

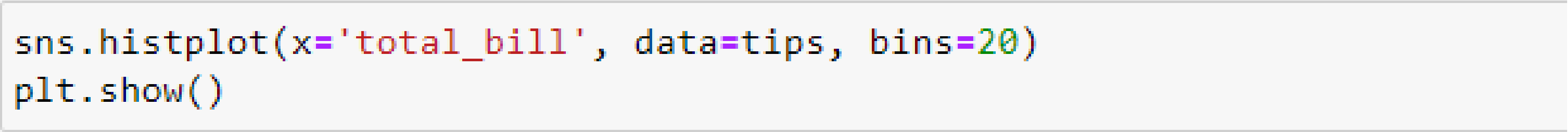


```
sns.boxplot(x="total_bill", y="day", data=tips, hue="sex")
plt.show()
```

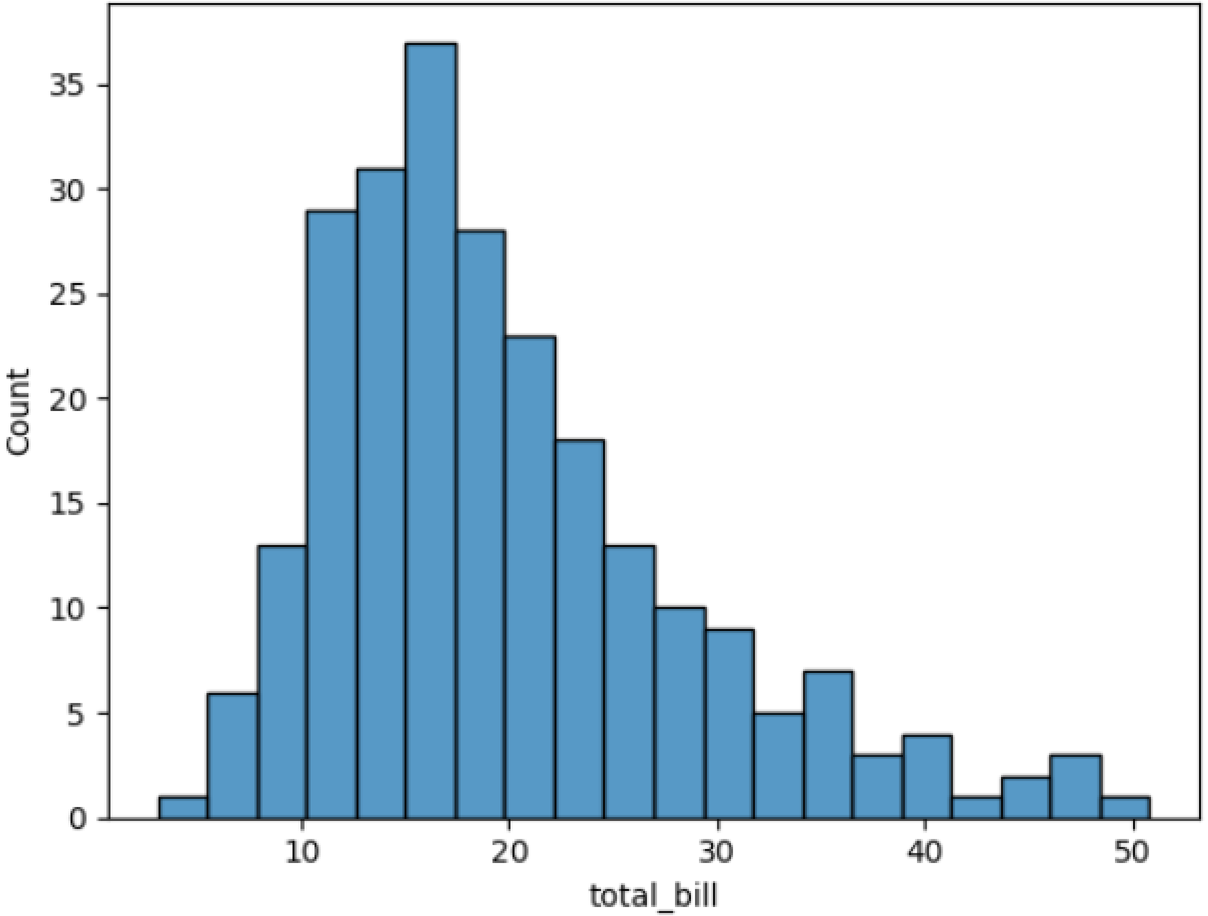


Seaborn

- **Histogram**



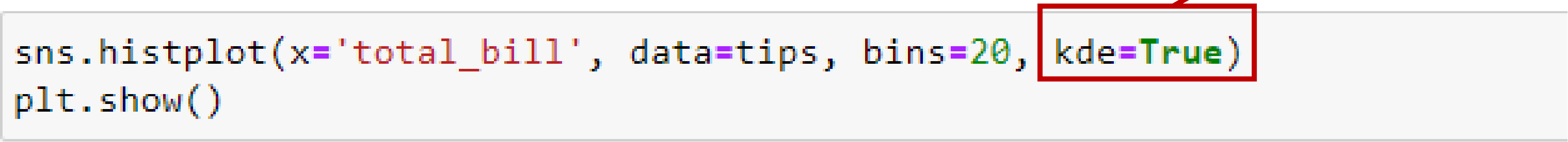
```
sns.histplot(x='total_bill', data=tips, bins=20)
plt.show()
```



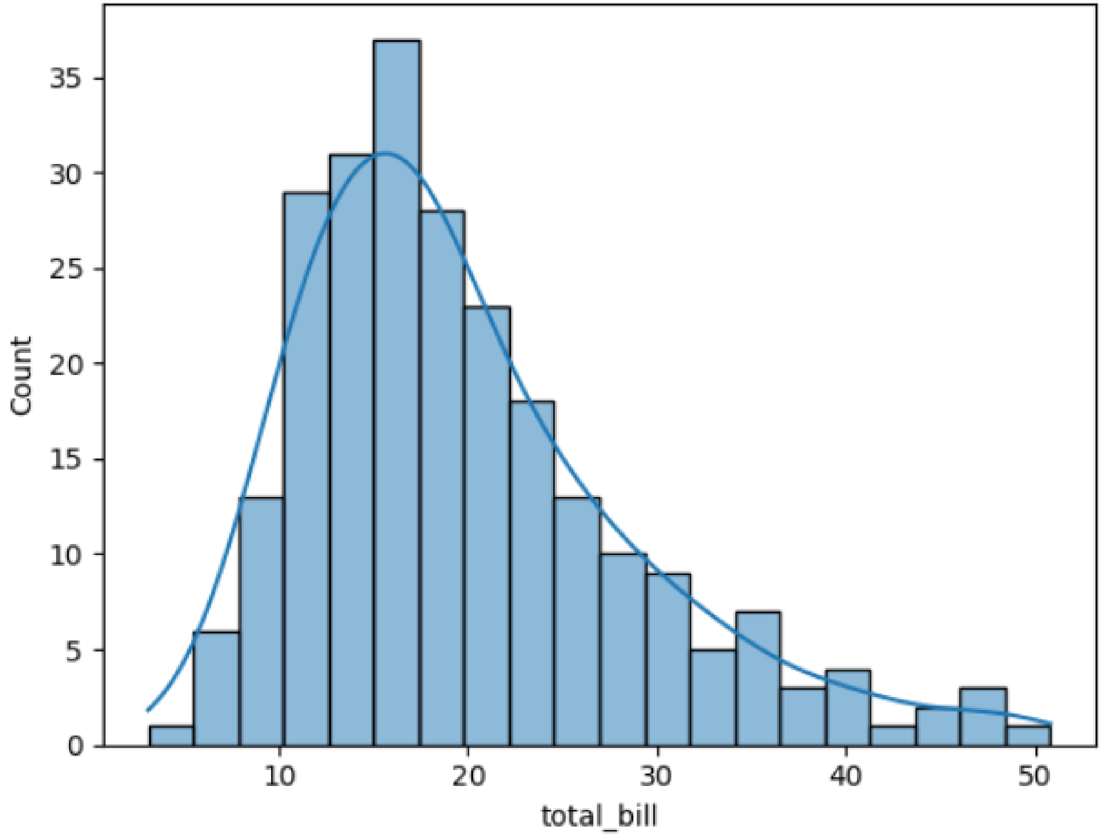
Seaborn

- Histogram

Kernel Density Estimation

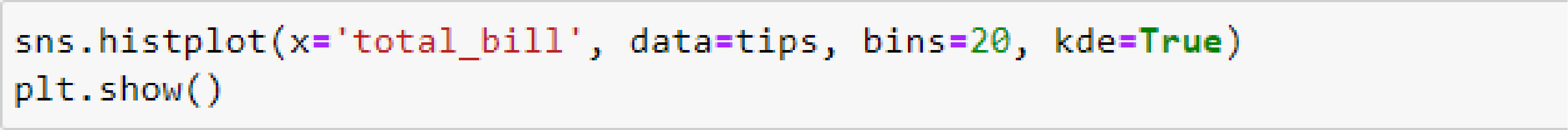


```
sns.histplot(x='total_bill', data=tips, bins=20, kde=True)
plt.show()
```

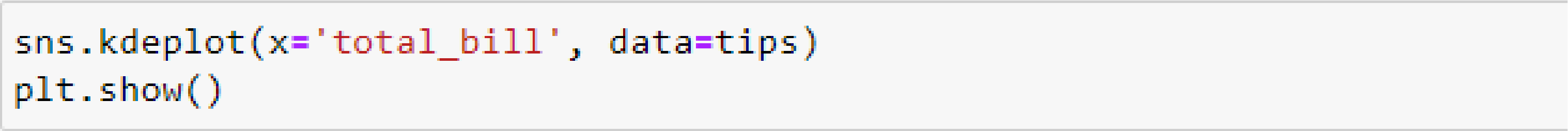


Seaborn

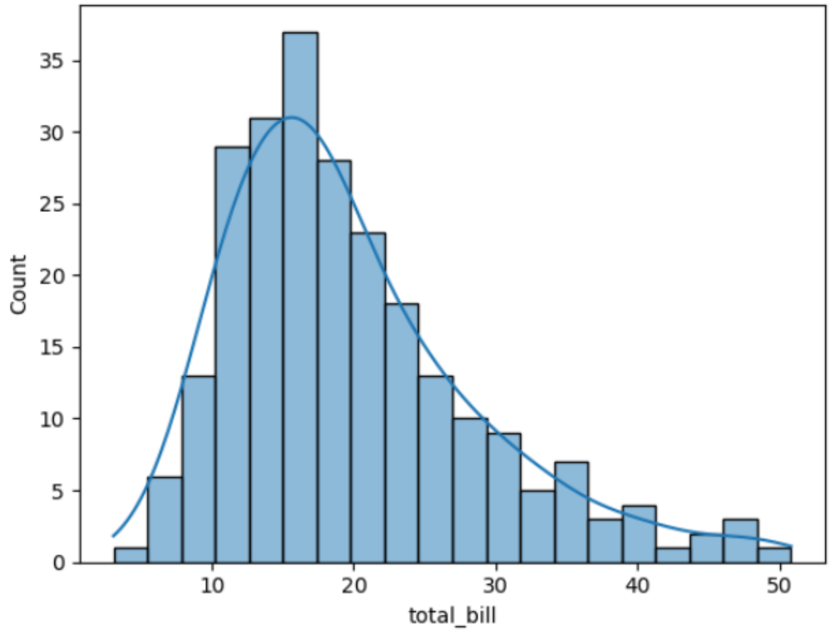
- **Histogram**

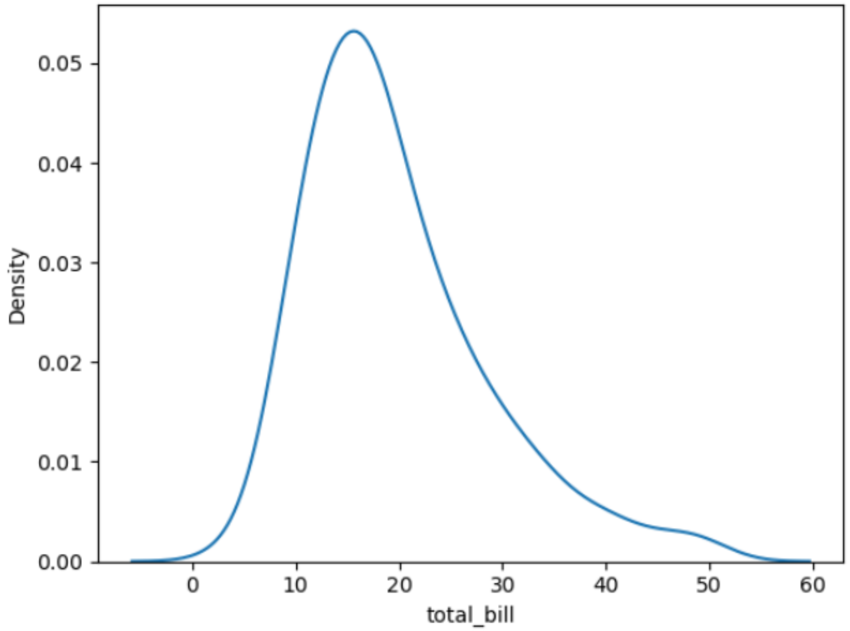


```
sns.histplot(x='total_bill', data=tips, bins=20, kde=True)
plt.show()
```



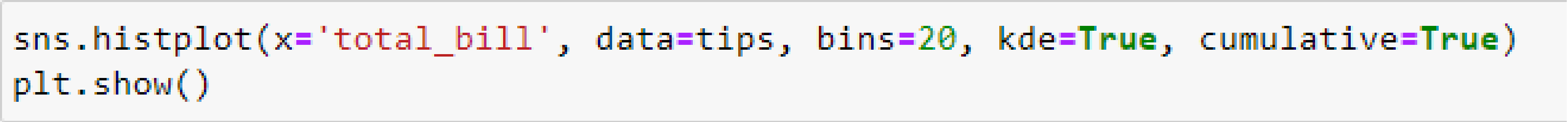
```
sns.kdeplot(x='total_bill', data=tips)
plt.show()
```



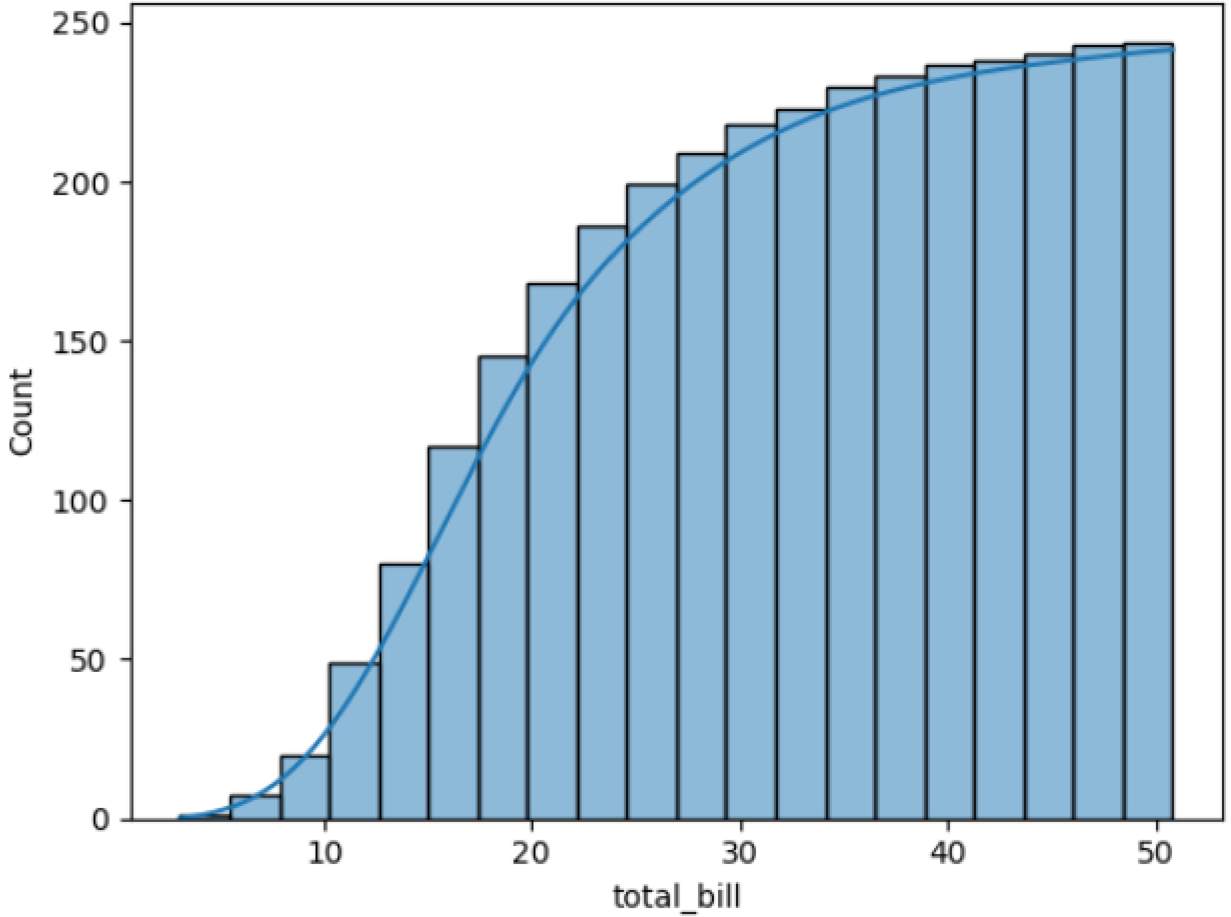


Seaborn

- **Histogram**

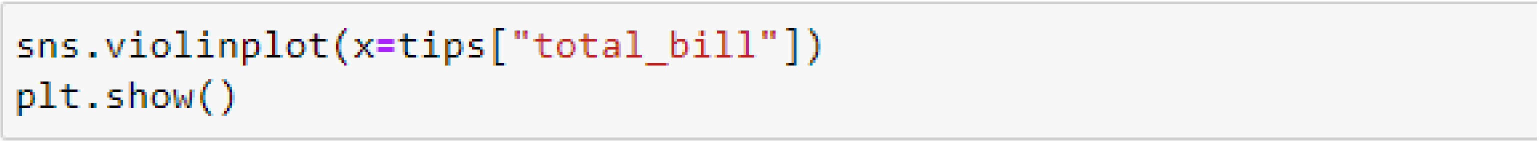


```
sns.histplot(x='total_bill', data=tips, bins=20, kde=True, cumulative=True)
plt.show()
```

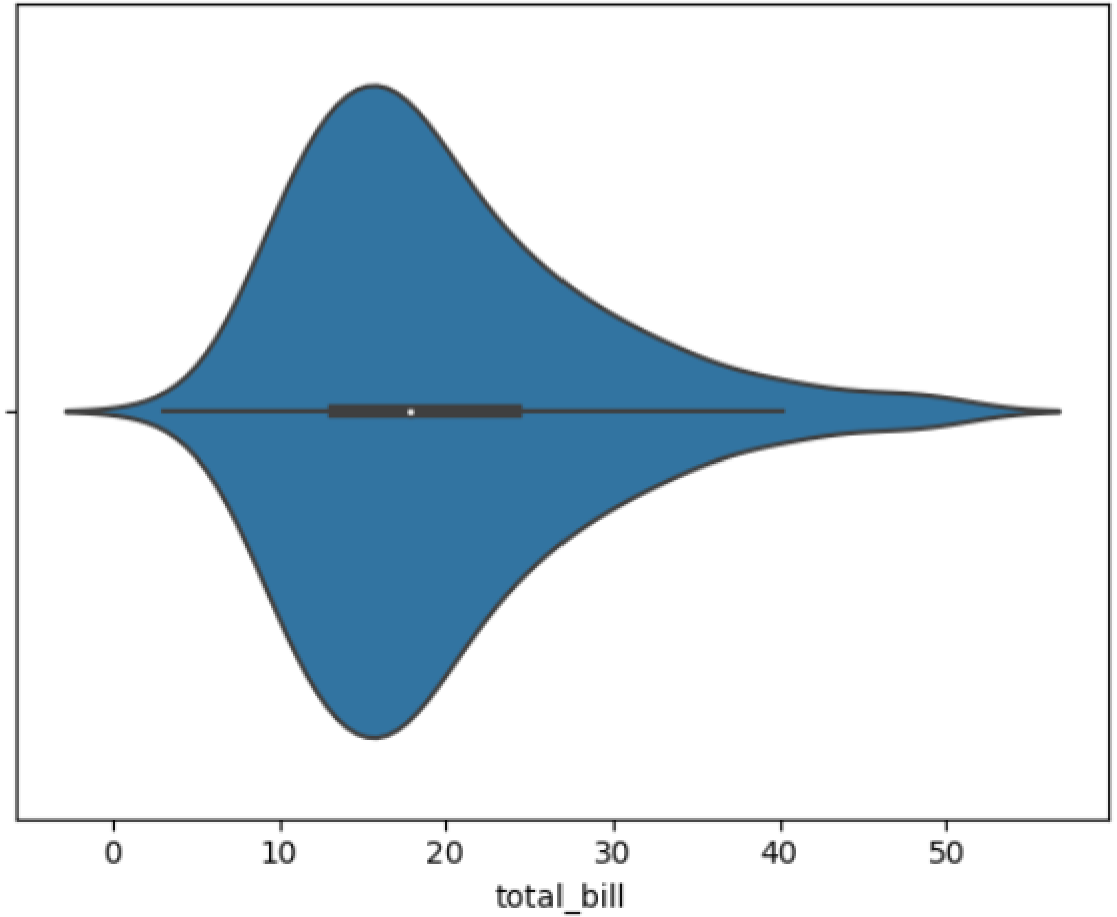


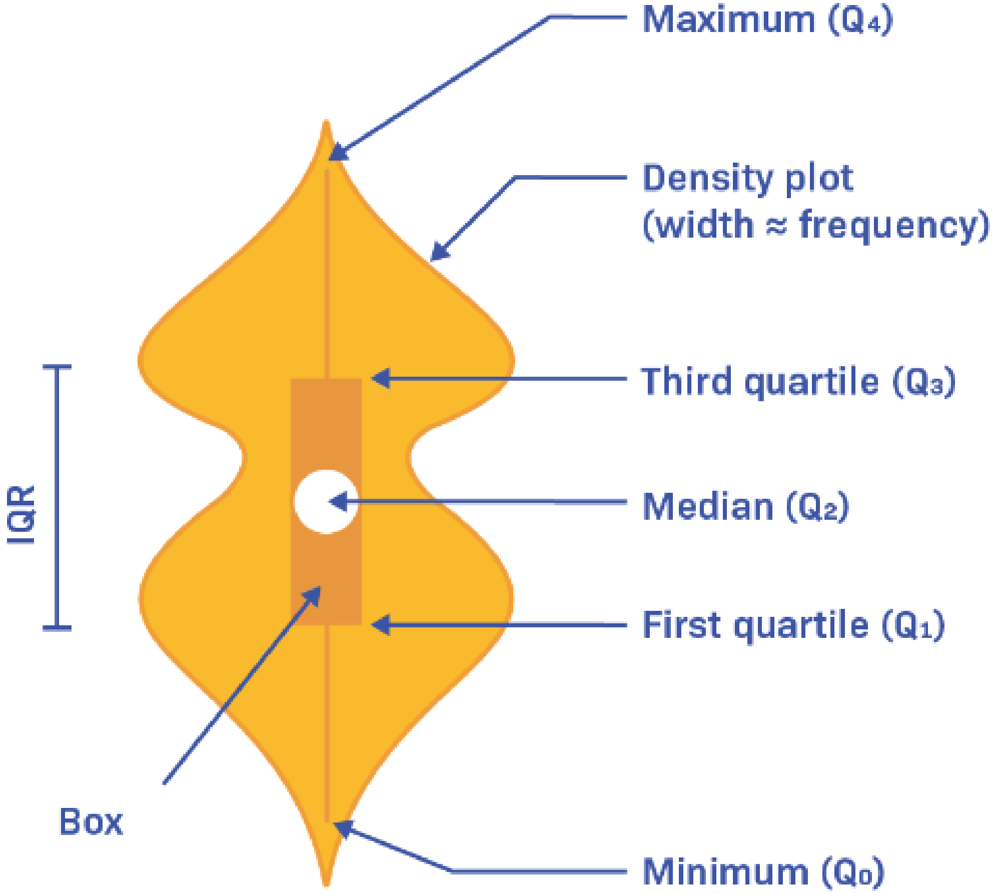
Seaborn

- **Violin Plot**



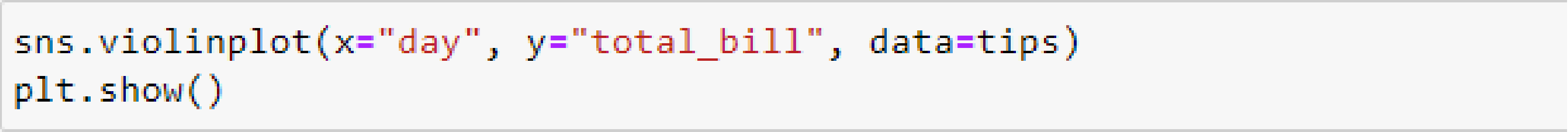
```
sns.violinplot(x=tips["total_bill"])
plt.show()
```



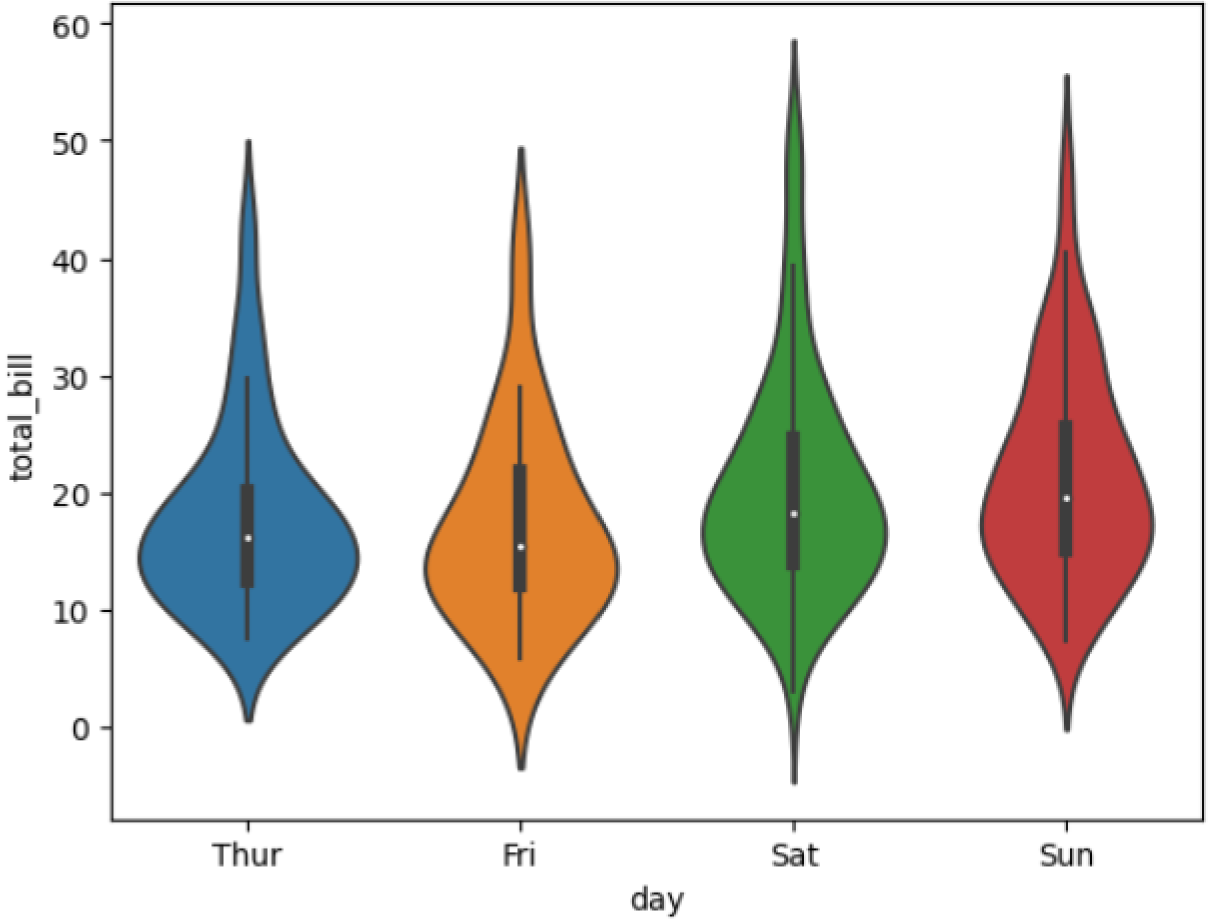


Seaborn

- **Violin Plot**

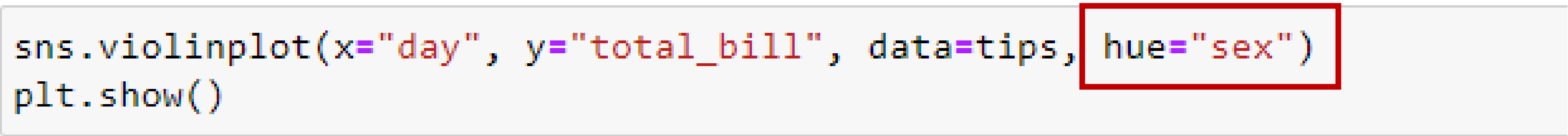


```
sns.violinplot(x="day", y="total_bill", data=tips)
plt.show()
```

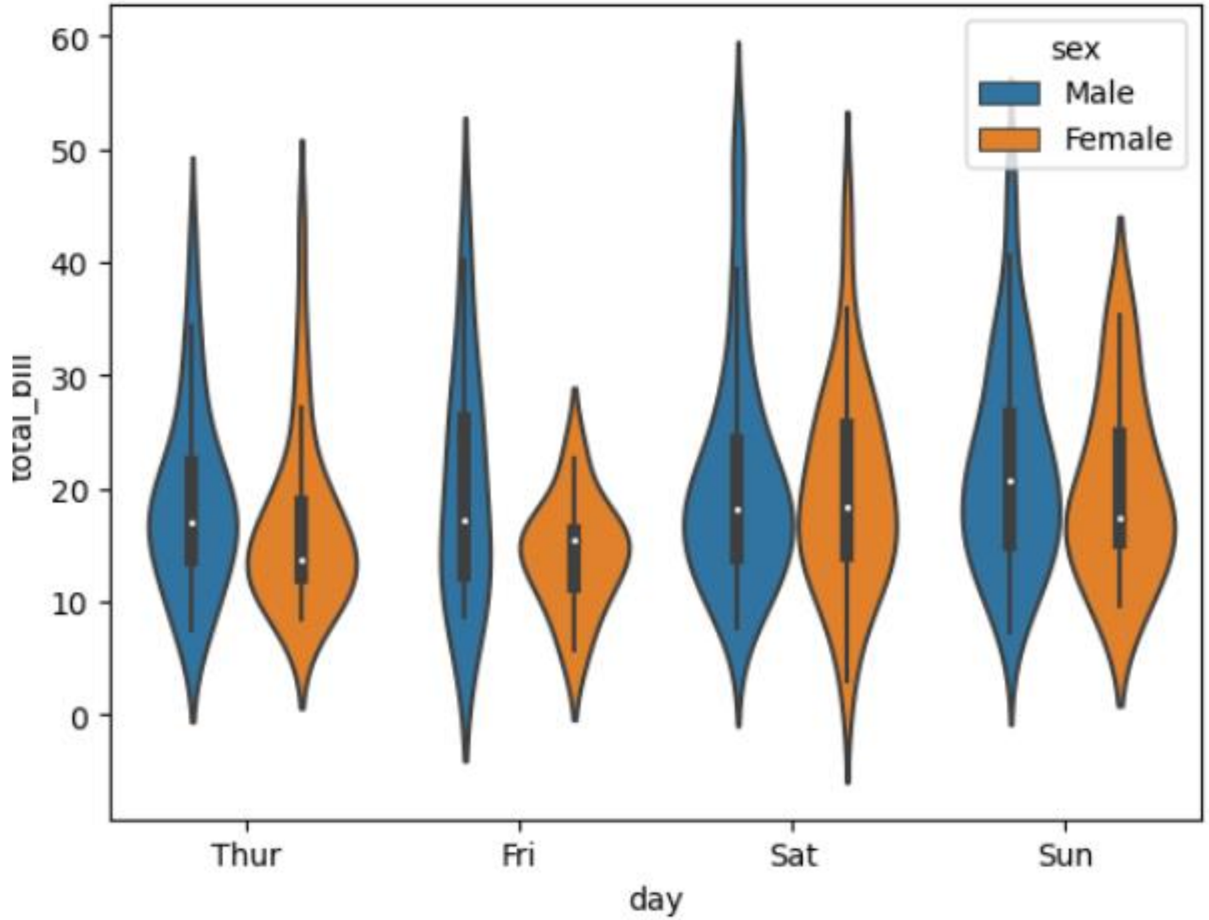


Seaborn

- **Violin Plot**

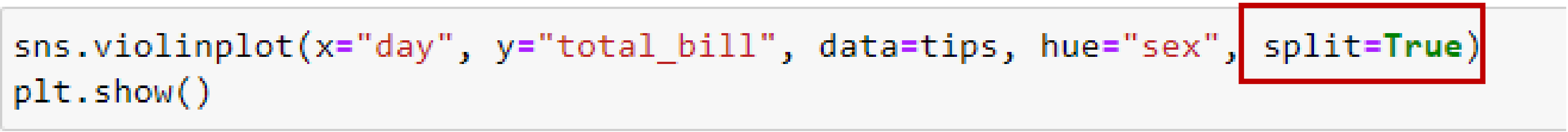


```
sns.violinplot(x="day", y="total_bill", data=tips, hue="sex")
plt.show()
```

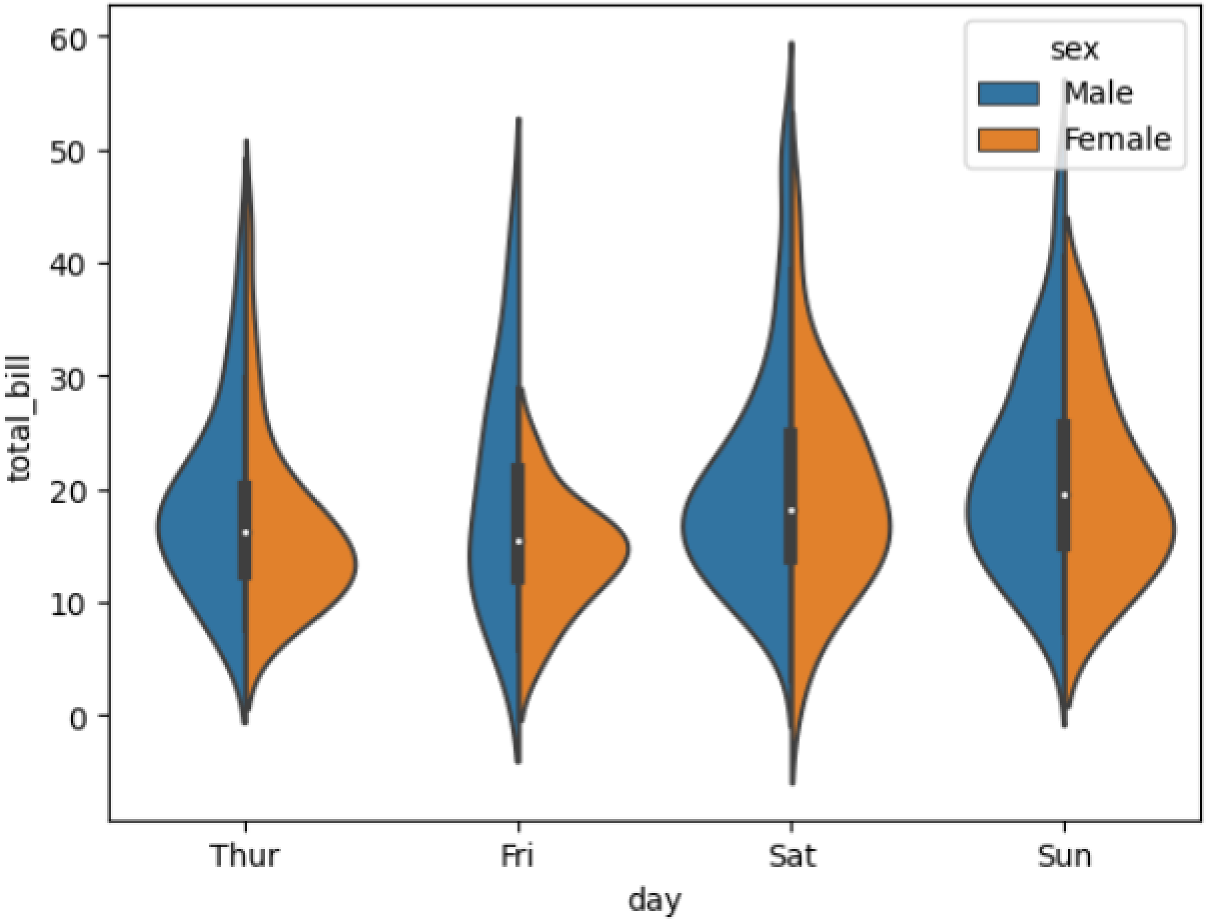


Seaborn

- **Violin Plot**

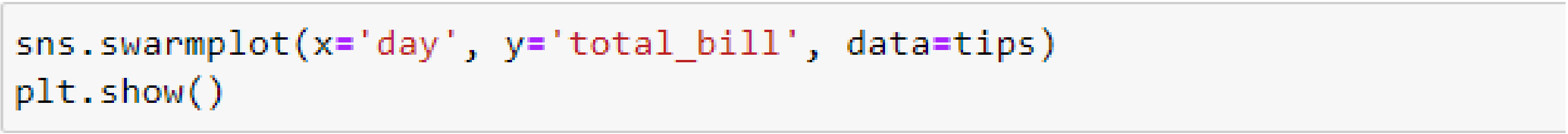


```
sns.violinplot(x="day", y="total_bill", data=tips, hue="sex", split=True)
plt.show()
```

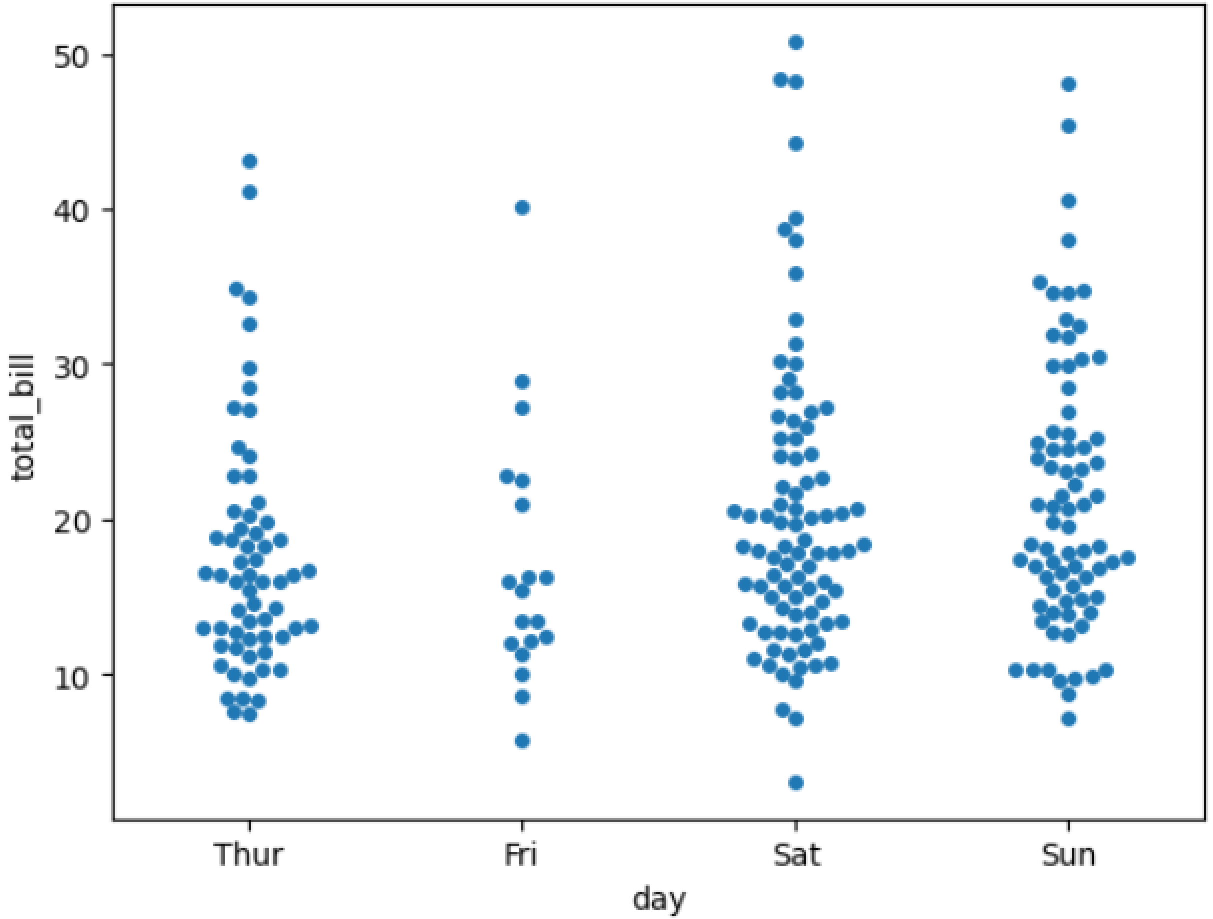


Seaborn

- **Swarm Plot**

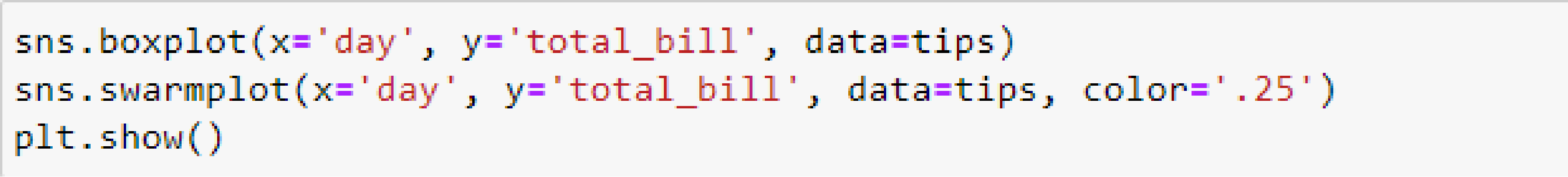


```
sns.swarmplot(x='day', y='total_bill', data=tips)
plt.show()
```

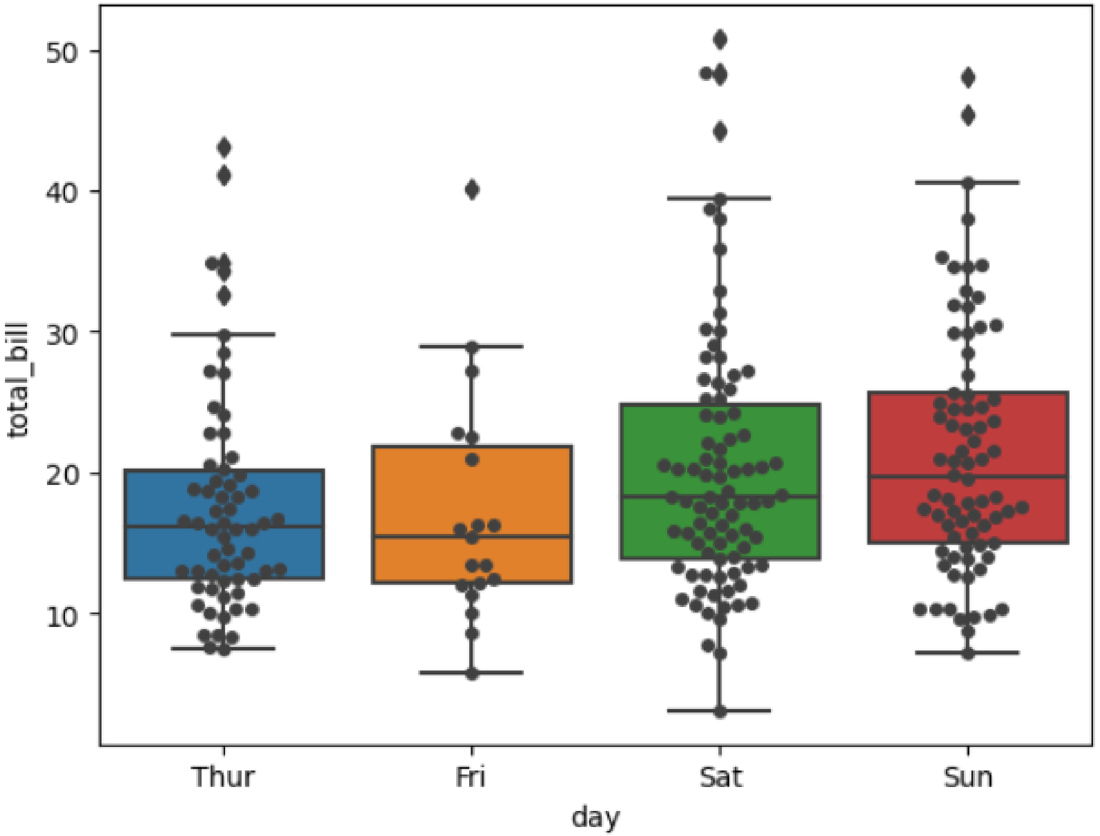


Seaborn

- **Swarm Plot**

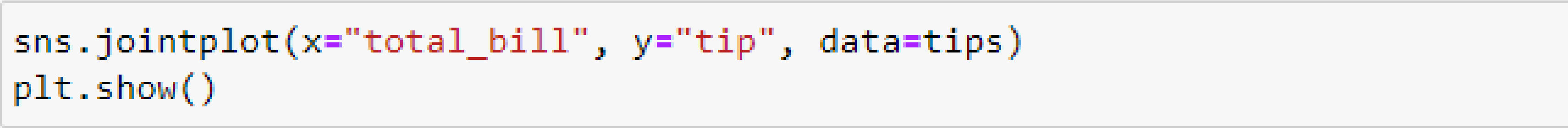


```
sns.boxplot(x='day', y='total_bill', data=tips)
sns.swarmplot(x='day', y='total_bill', data=tips, color='.25')
plt.show()
```

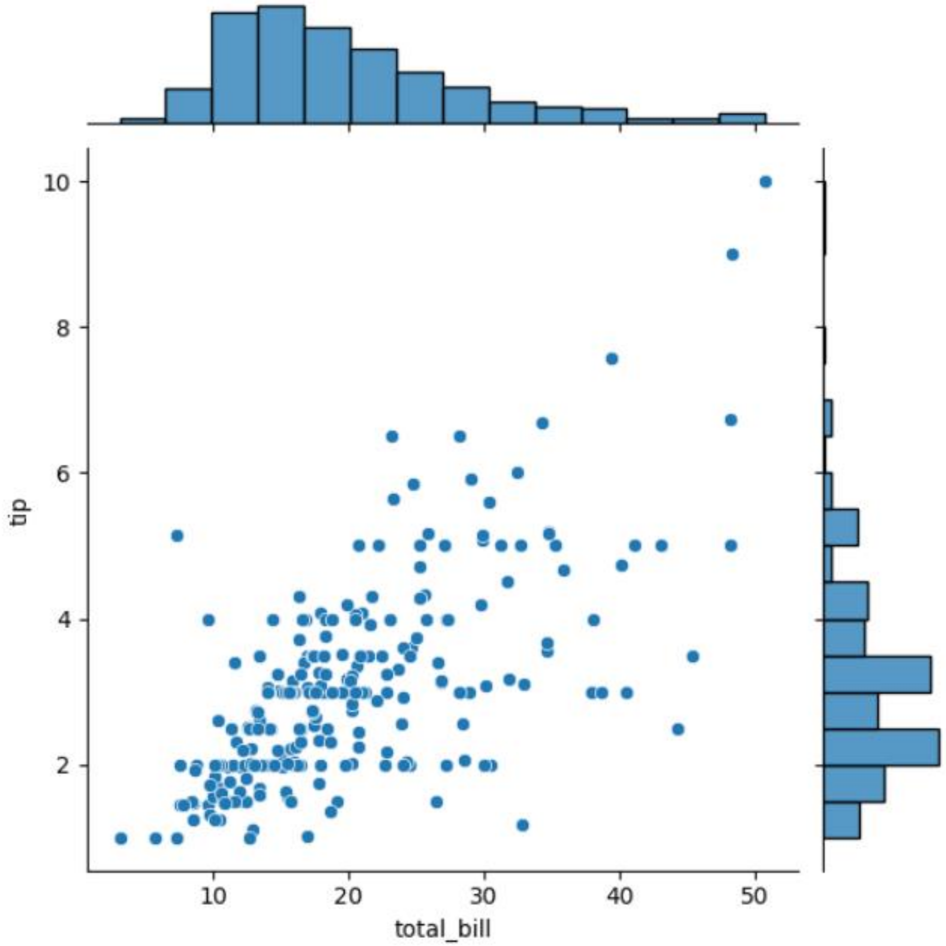


Seaborn

- ## Joint Plot

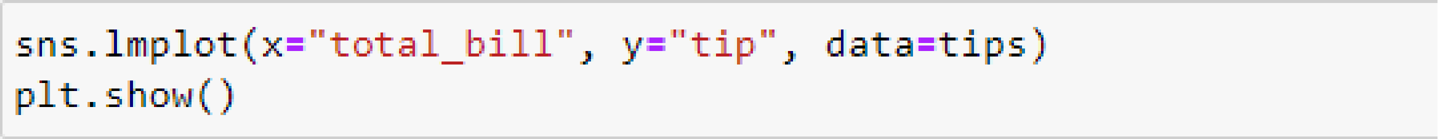


```
sns.jointplot(x="total_bill", y="tip", data=tips)
plt.show()
```

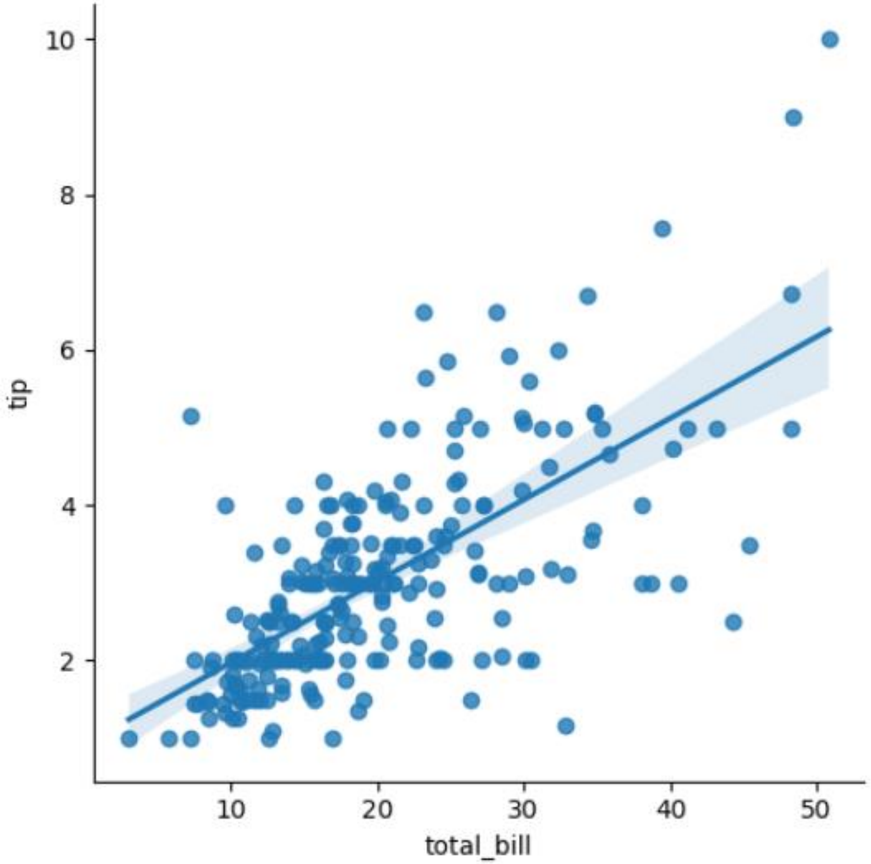


Seaborn

- **Linear Model Plot**

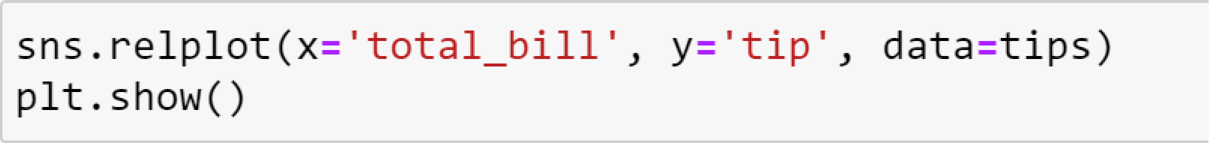


```
sns.lmplot(x="total_bill", y="tip", data=tips)
plt.show()
```

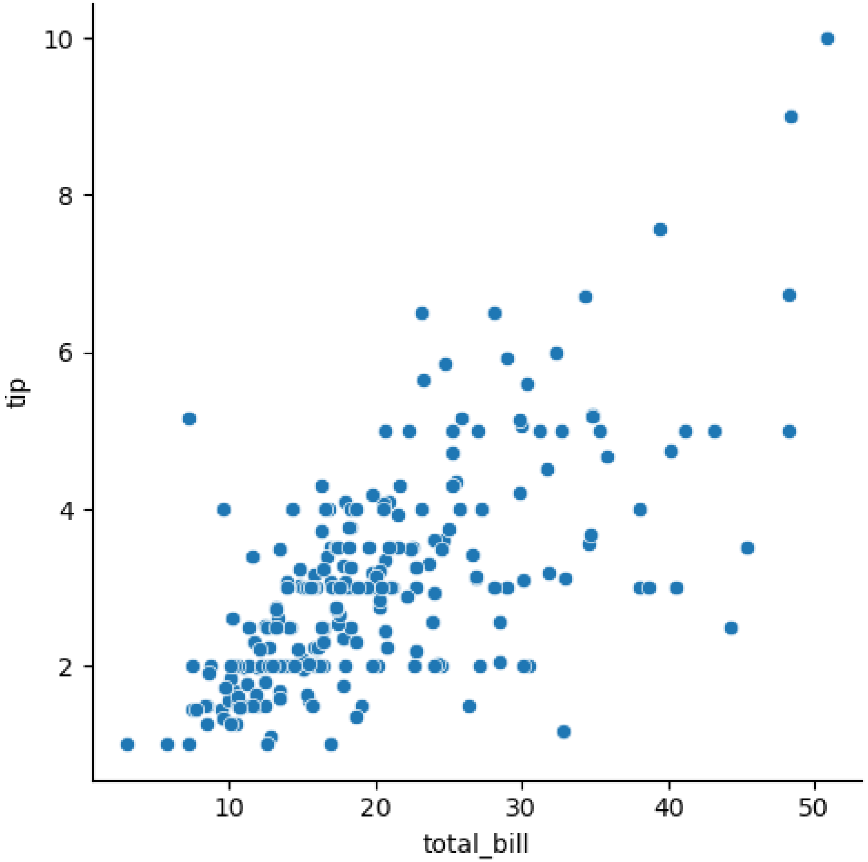


Seaborn

- **Relation Plot**

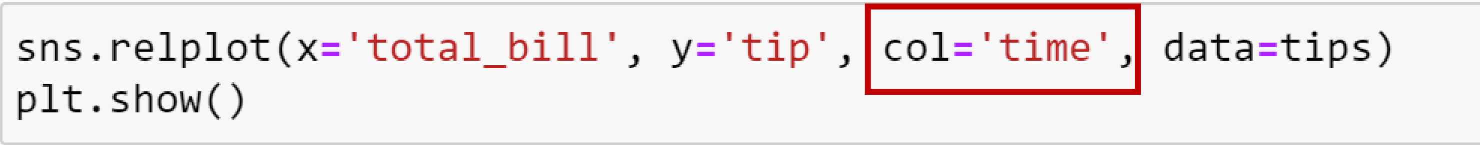


```
sns.relplot(x='total_bill', y='tip', data=tips)
plt.show()
```

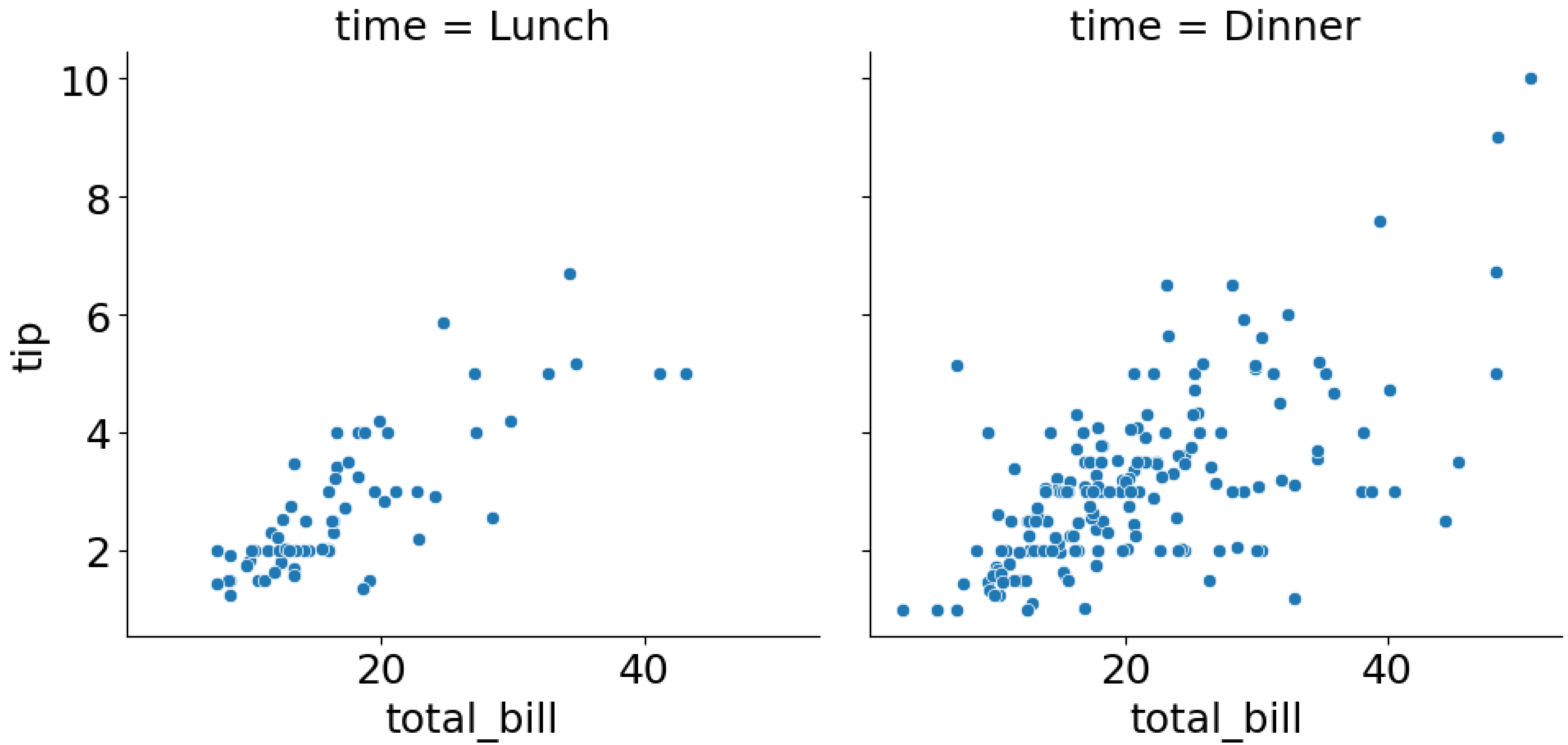


Seaborn

- **Relation Plot**

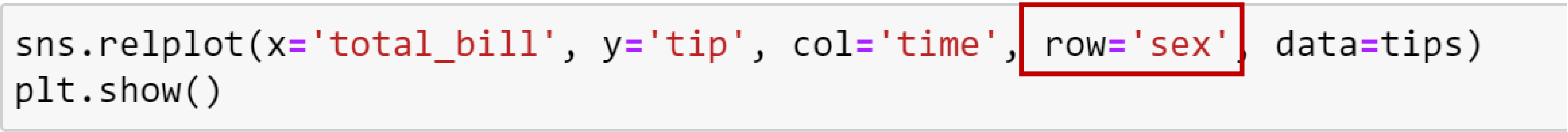


```
sns.relplot(x='total_bill', y='tip', col='time', data=tips)
plt.show()
```

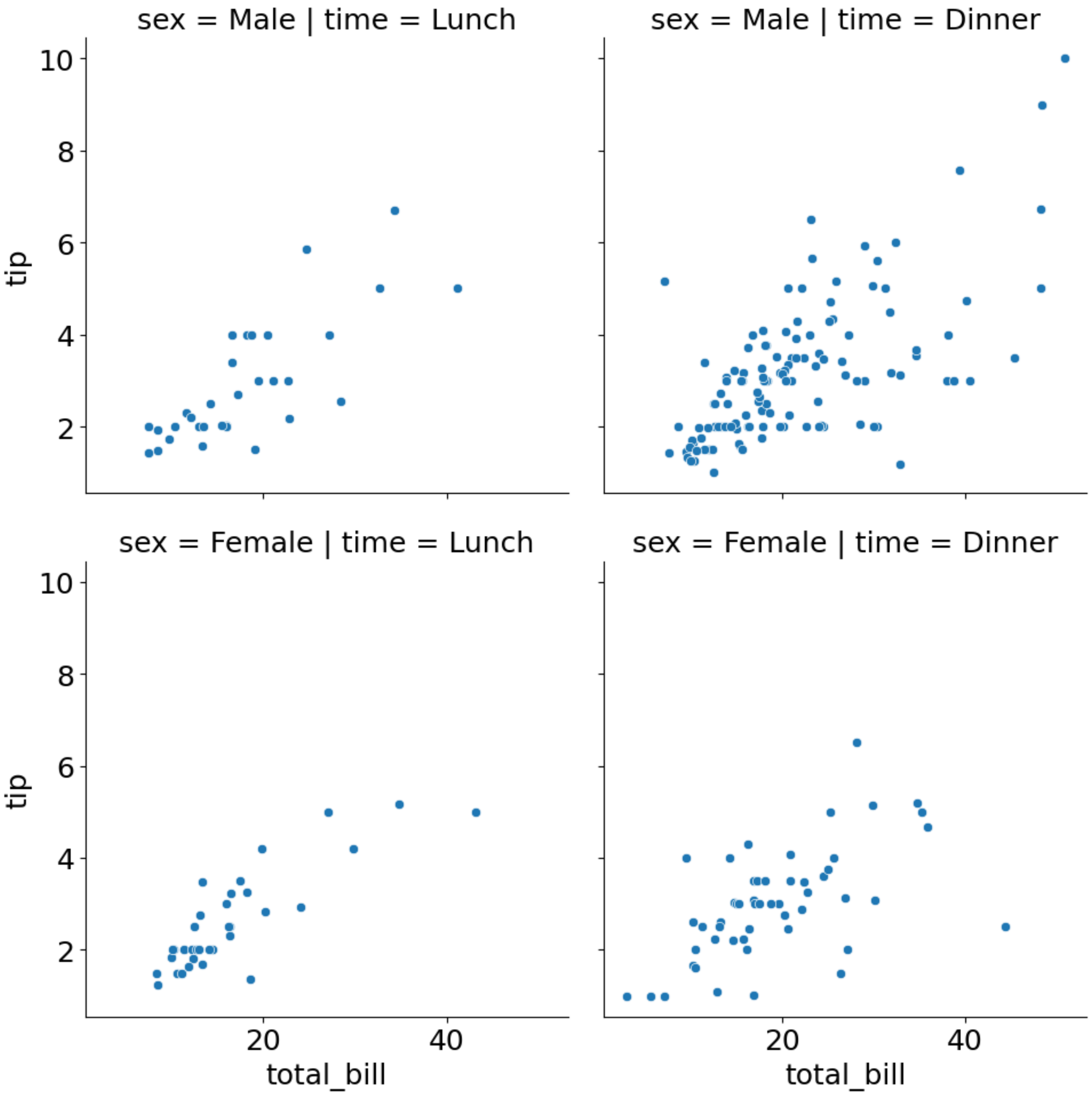


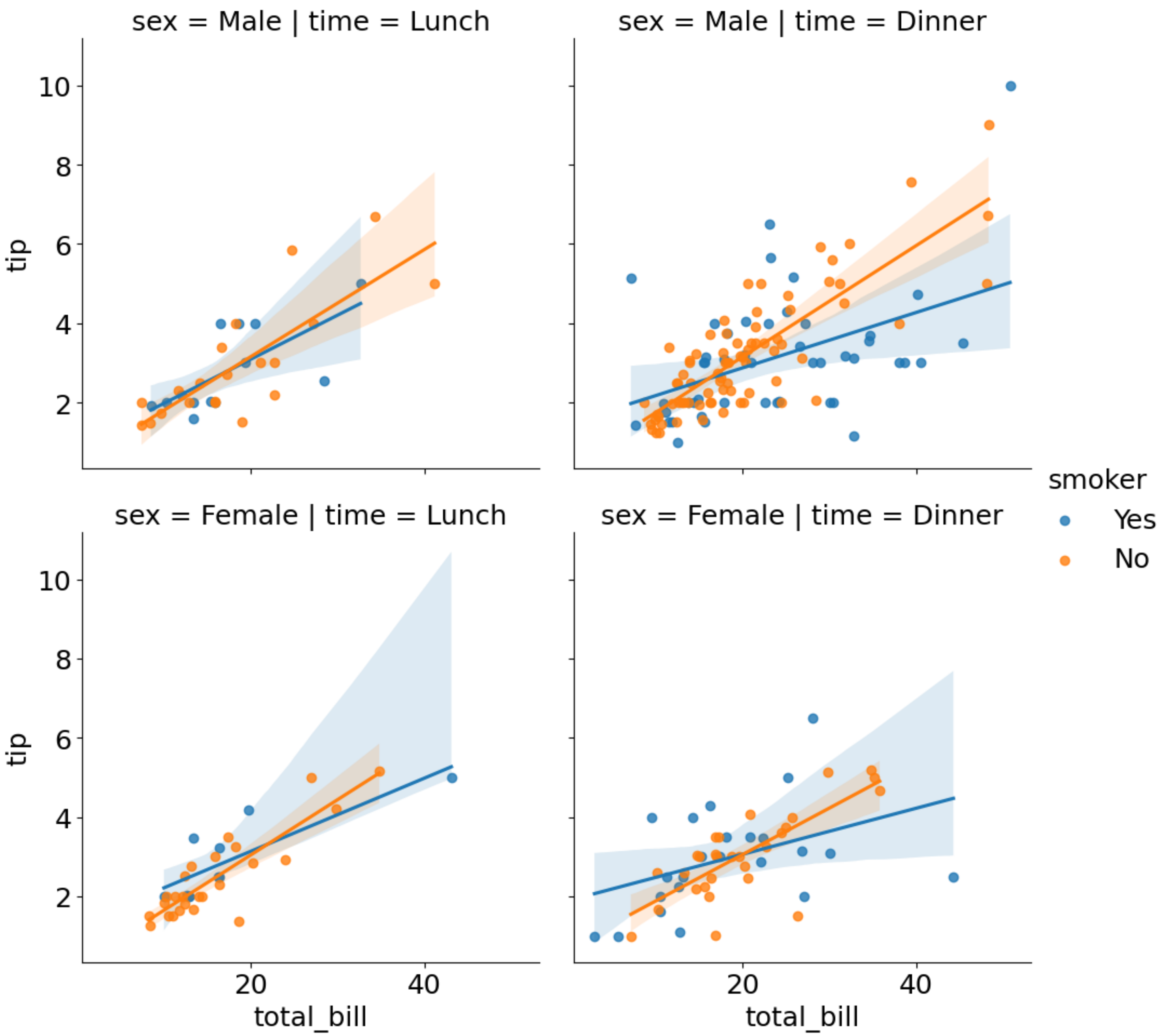
Seaborn

- **Relation Plot**



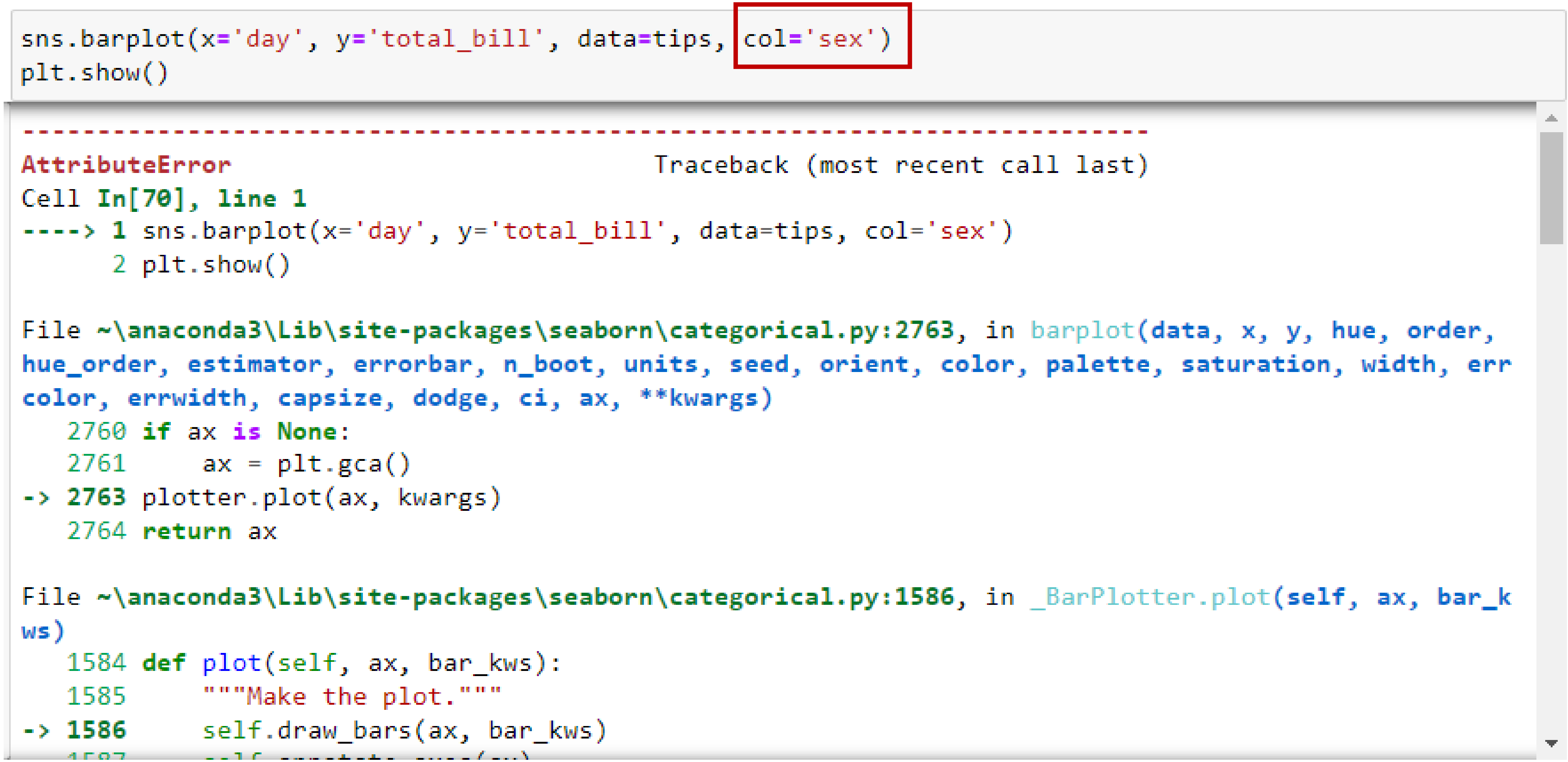
```
sns.relplot(x='total_bill', y='tip', col='time', row='sex', data=tips)
plt.show()
```


sex = Male | time = Lunch
sex = Male | time = Dinner
sex = Female | time = Lunch
sex = Female | time = Dinner
tip
10
8
6
4
2
tip
10
8
6
4
2
20
40
total_bill
20
40
total_bill


sex = Male | time = Lunch
sex = Male | time = Dinner
sex = Female | time = Lunch
sex = Female | time = Dinner
tip
tip
10
8
6
4
2
20
40
total_bill
total_bill
smoker
Yes
No

Seaborn

- **Categorical Plot**

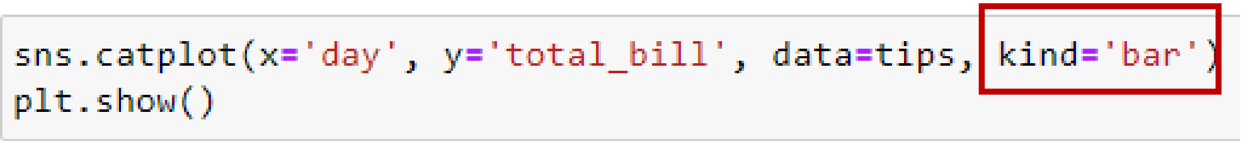


```
sns.barplot(x='day', y='total_bill', data=tips, col='sex')
plt.show()
```

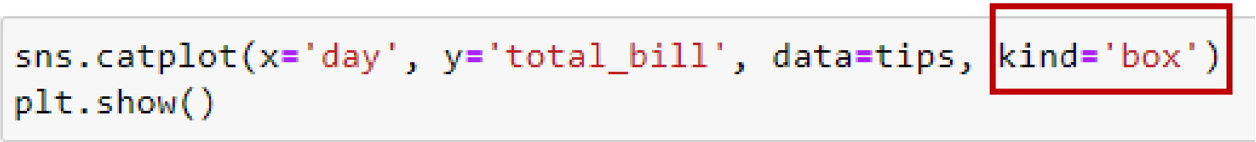
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[70], line 1
----> 1 sns.barplot(x='day', y='total_bill', data=tips, col='sex')
      2 plt.show()

File ~\anaconda3\Lib\site-packages\seaborn\categorical.py:2763, in barplot(data, x, y, hue, order,
hue_order, estimator, errorbar, n_boot, units, seed, orient, color, palette, saturation, width, err
color, errwidth, capsize, dodge, ci, ax, **kwargs)
   2760 if ax is None:
   2761     ax = plt.gca()
-> 2763 plotter.plot(ax, kwargs)
   2764 return ax

File ~\anaconda3\Lib\site-packages\seaborn\categorical.py:1586, in _BarPlotter.plot(self, ax, bar_k
ws)
   1584 def plot(self, ax, bar_kws):
   1585     """Make the plot."""
-> 1586     self.draw_bars(ax, bar_kws)
   1587     self._annotate_axes(ax)
```

Seaborn

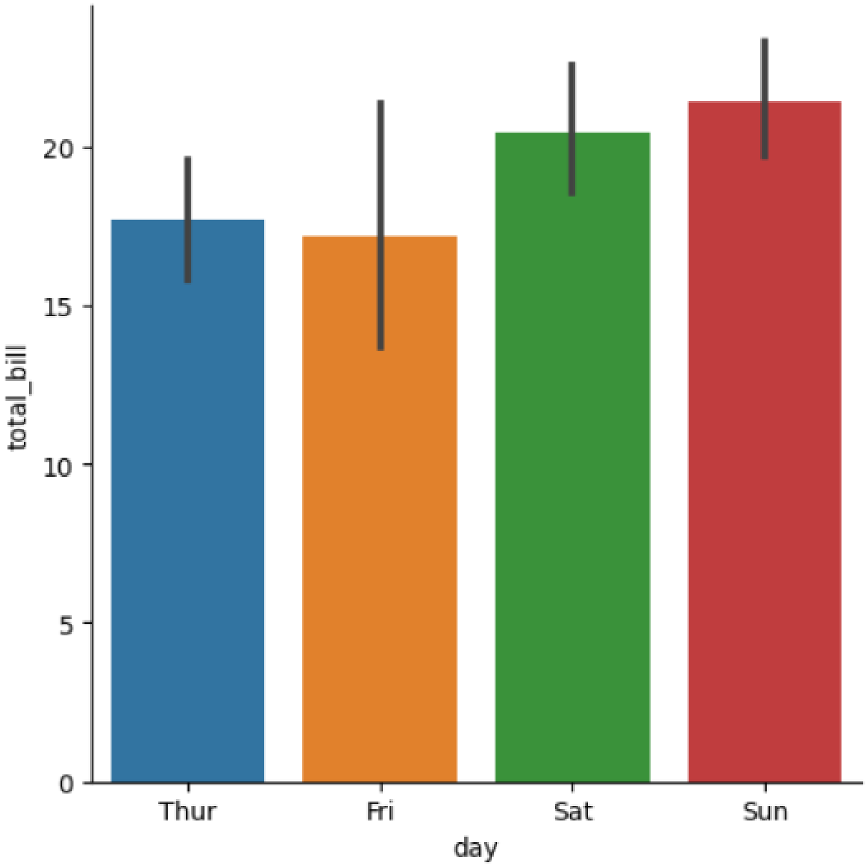
- **Categorical Plot**

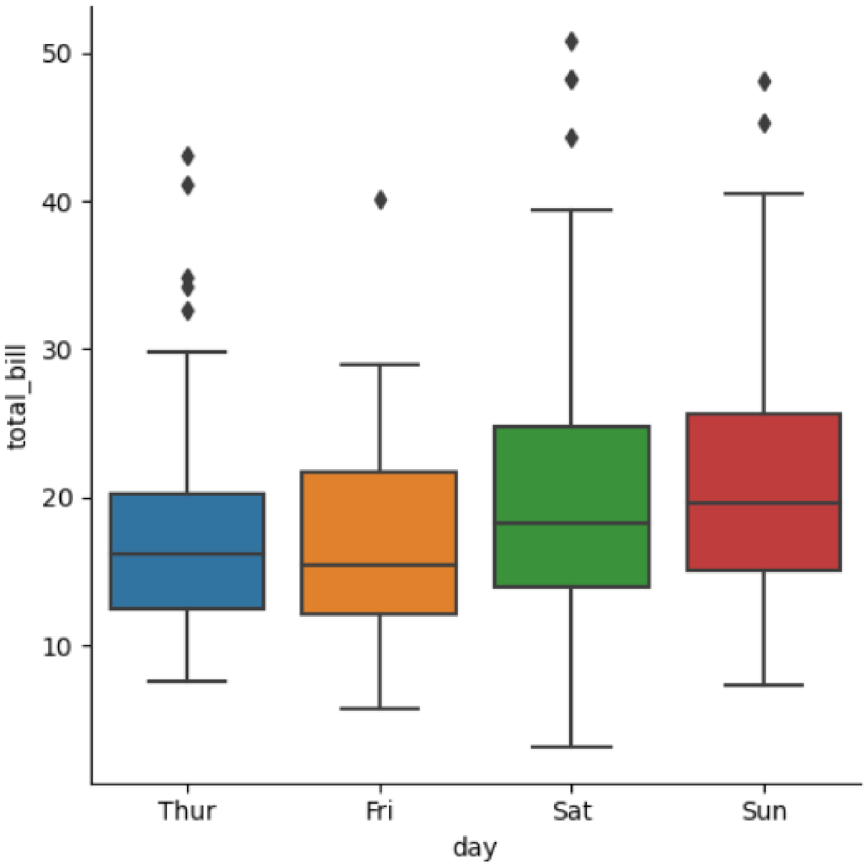


```
sns.catplot(x='day', y='total_bill', data=tips, kind='bar')
plt.show()
```



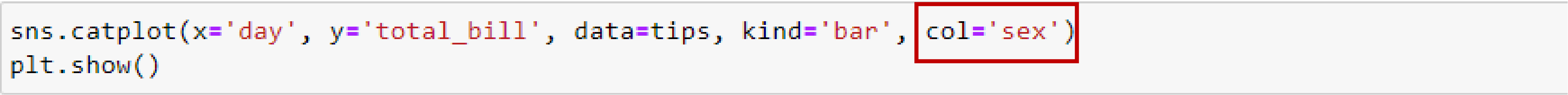
```
sns.catplot(x='day', y='total_bill', data=tips, kind='box')
plt.show()
```



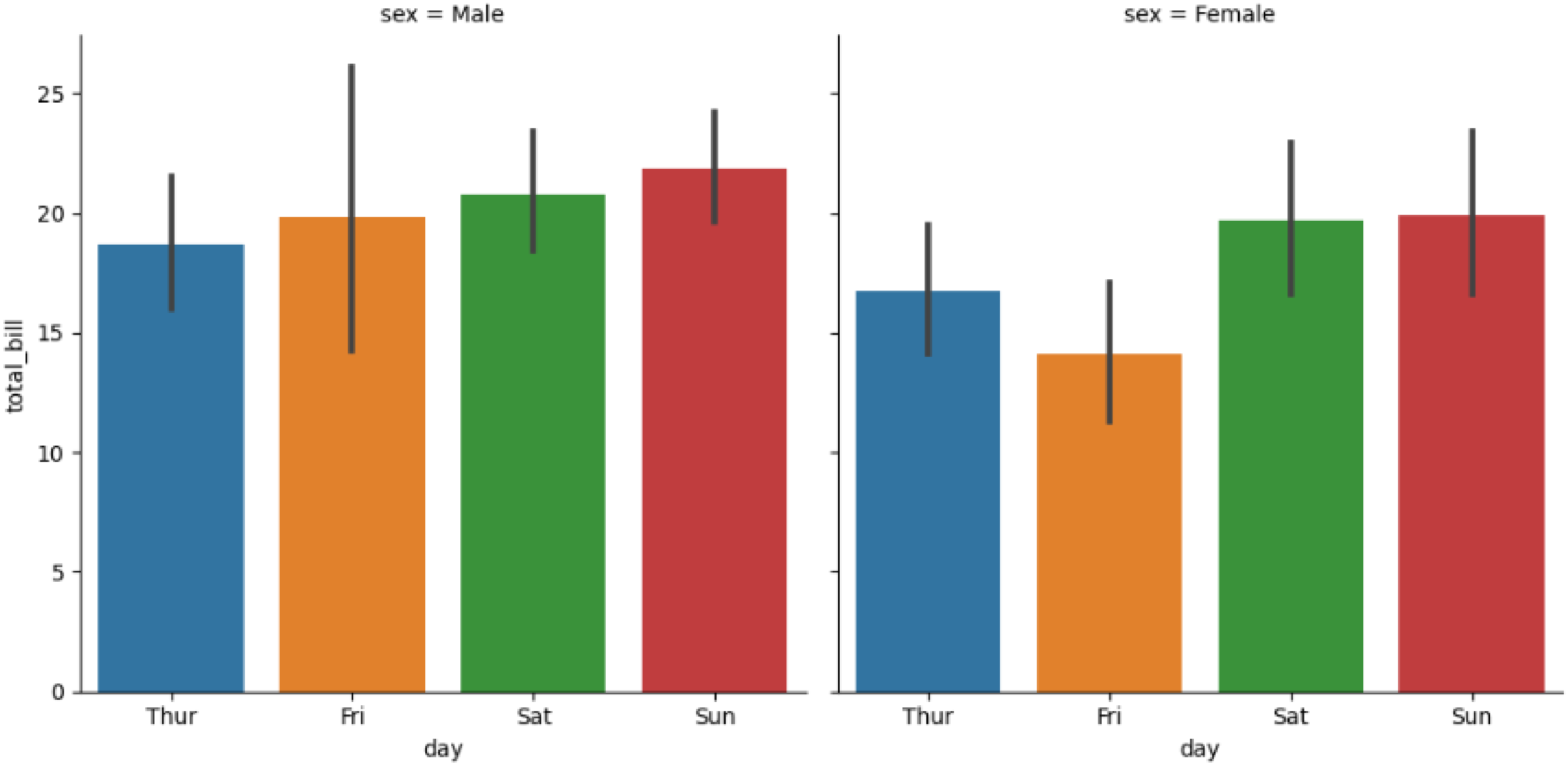


Seaborn

- **Categorical Plot**



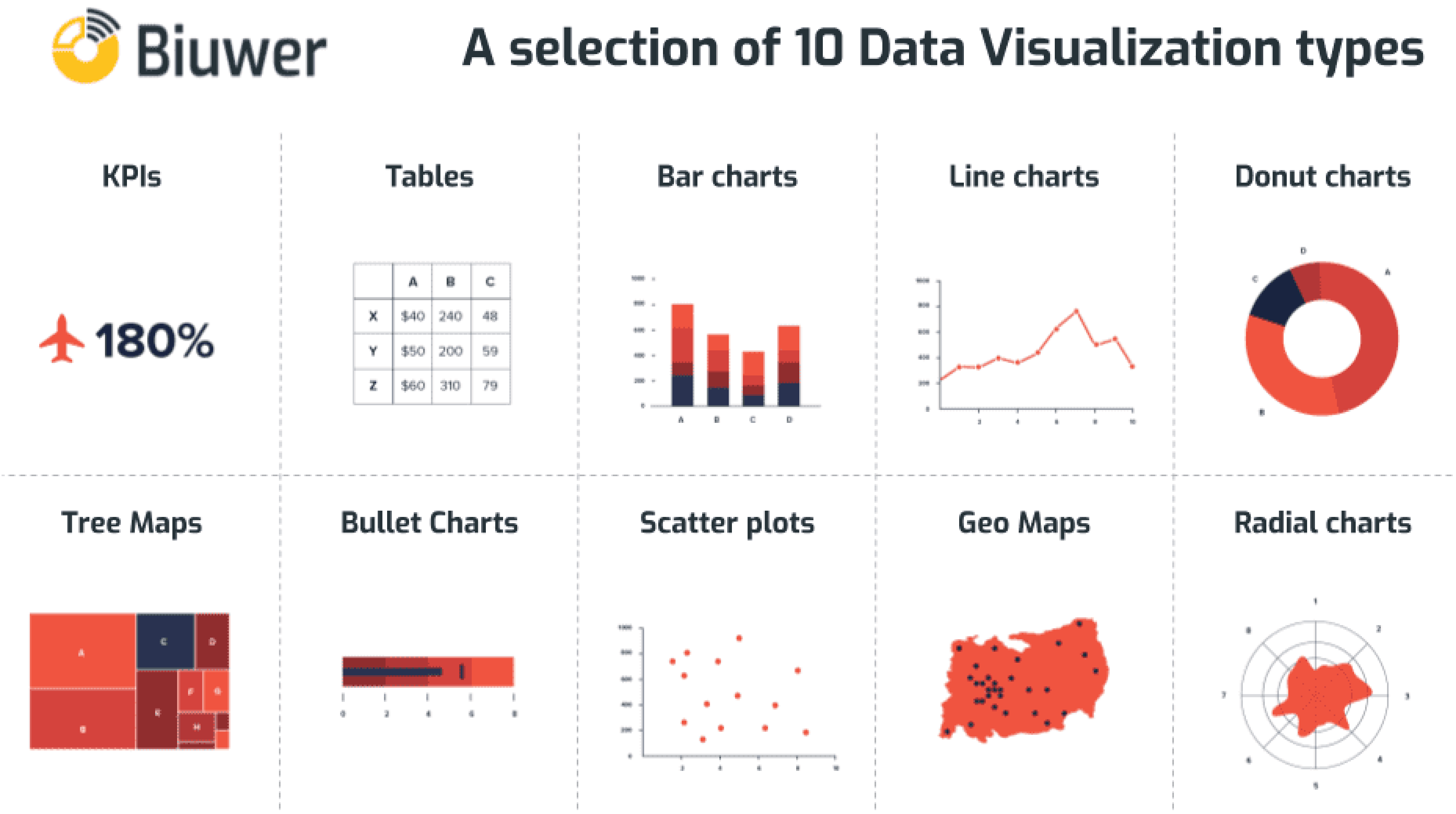
```
sns.catplot(x='day', y='total_bill', data=tips, kind='bar', col='sex')
plt.show()
```



Seaborn

seaborn
Installing
Gallery
Tutorial
API
Releases
Citing
FAQ
species
setosa
versicol
species
Adelie
Chinstrap
Gentoo
walk = 4
walk = 5
walk = 8
walk = 9
45°
135°
0°
180°
315°
225°

Geospatial Data Visualization



Biuwer

A selection of 10 Data Visualization types

KPIs

Tables

	A	B	C
X	$40	240	48
Y	$50	200	59
Z	$60	310	79

Bar charts

Line charts

Donut charts

Tree Maps

Bullet Charts

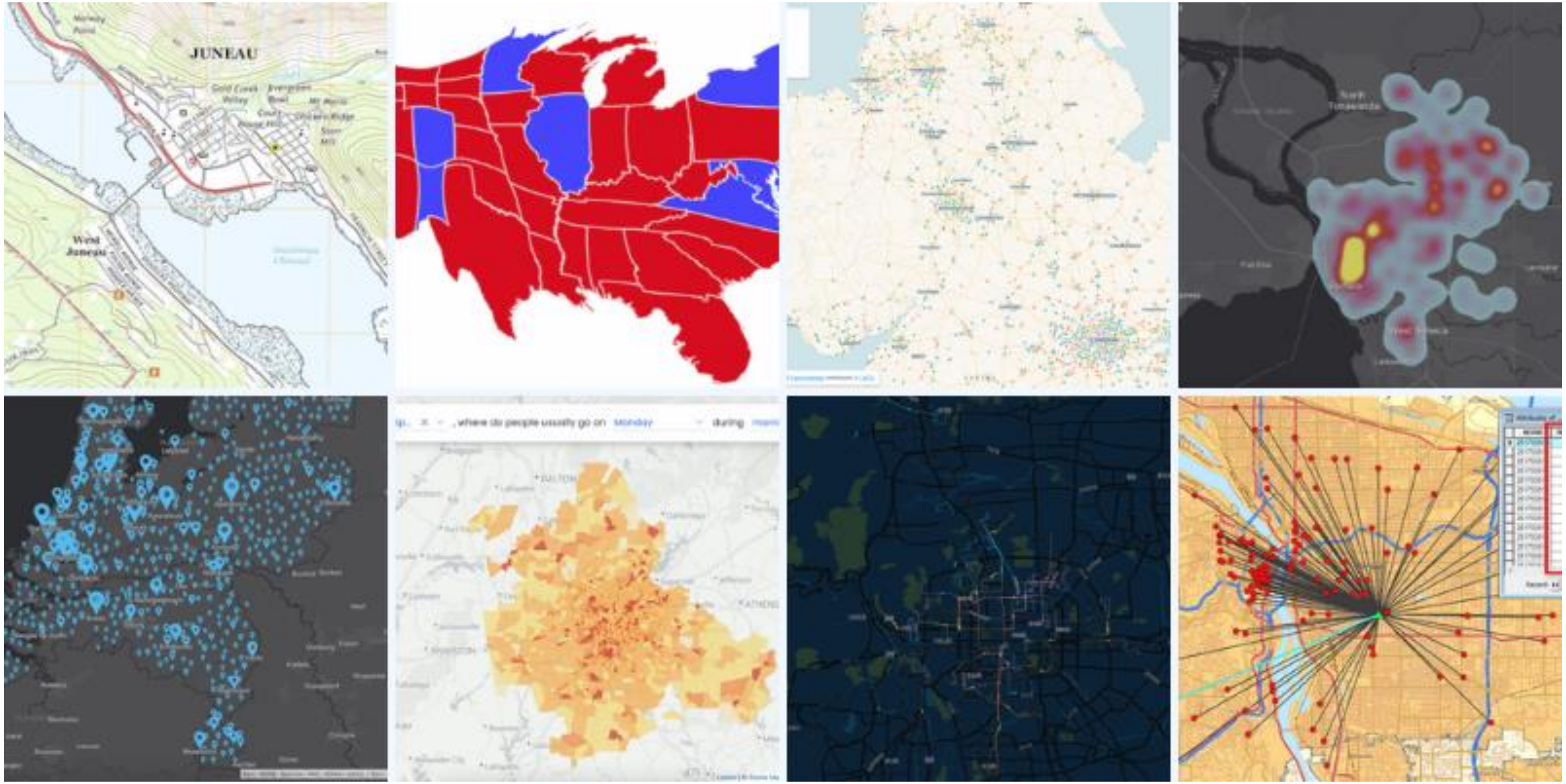
Scatter plots

Geo Maps

Radial charts

Geospatial Data Visualization

- **Geospatial Data Visualization**
  - Ability to visualize location related information easily, and improve insights to foster decisions



Week 14 Assignment

- Assignment (1) Drawing charts: Using Matplotlib

- Assignment (2) Drawing charts: Using Seaborn

Week 14 Assignment

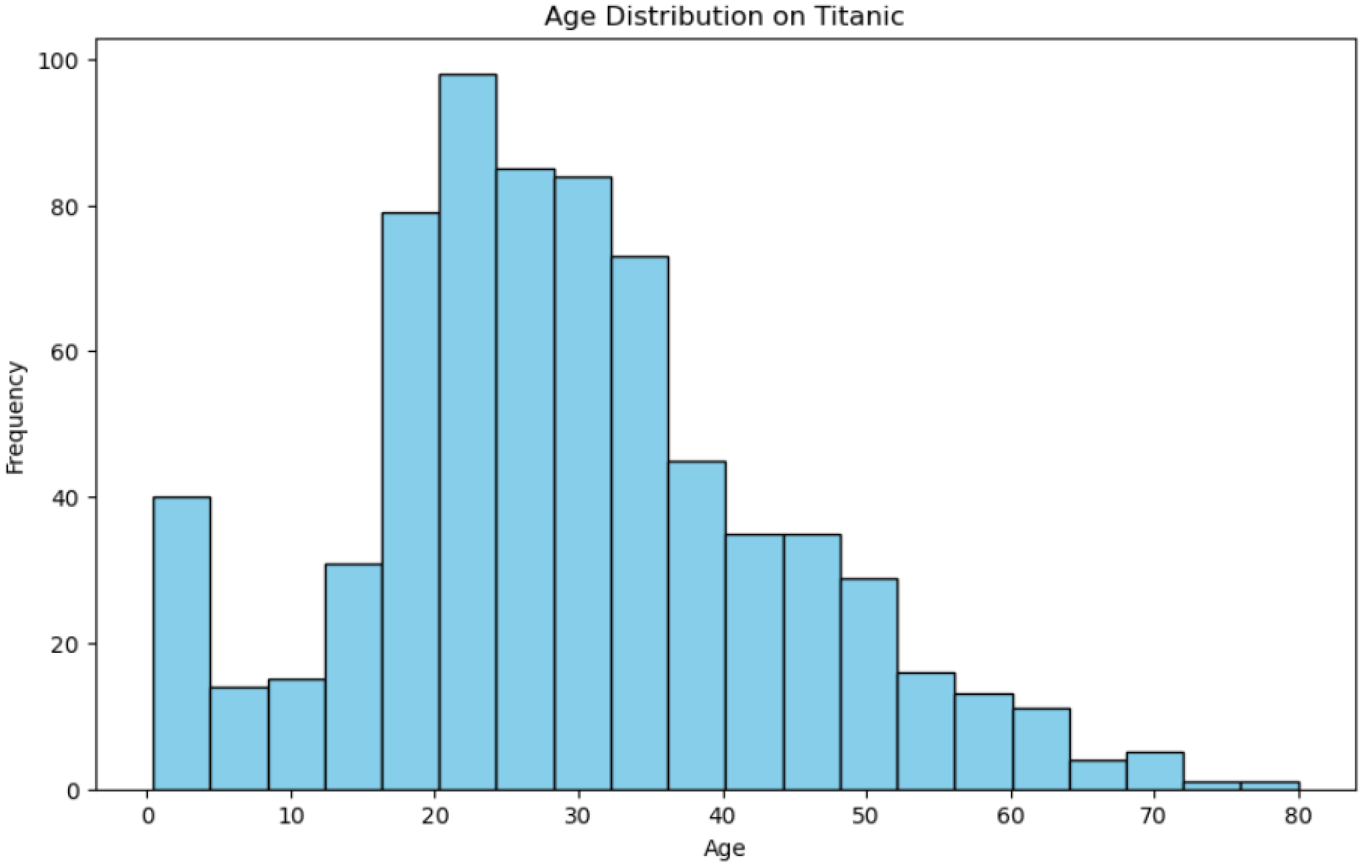
- **Submission due :** June 16th, 23:55
- **What to submit :** Notebook file (.ipynb) *** Submit each assignment as a separate file**
  - Colab : [File]-[Download]-[Download .ipynb]
  - Kaggle : [File]-[Download Notebook]

- **IMPORTANT**
  - Using the **matplotlib** library for Assignment (1)
  - Using the **seaborn** library for Assignment (2)
  - The design of the graph such as color or width does not need to be the same
  - The type of graph must be the same
  - For Assignment (1), Be sure to download the dataset from Assignment Week14
    - The file name is “titanic.csv”.
    - You don’t need to clean the dataset

Week 14 Assignment (1)

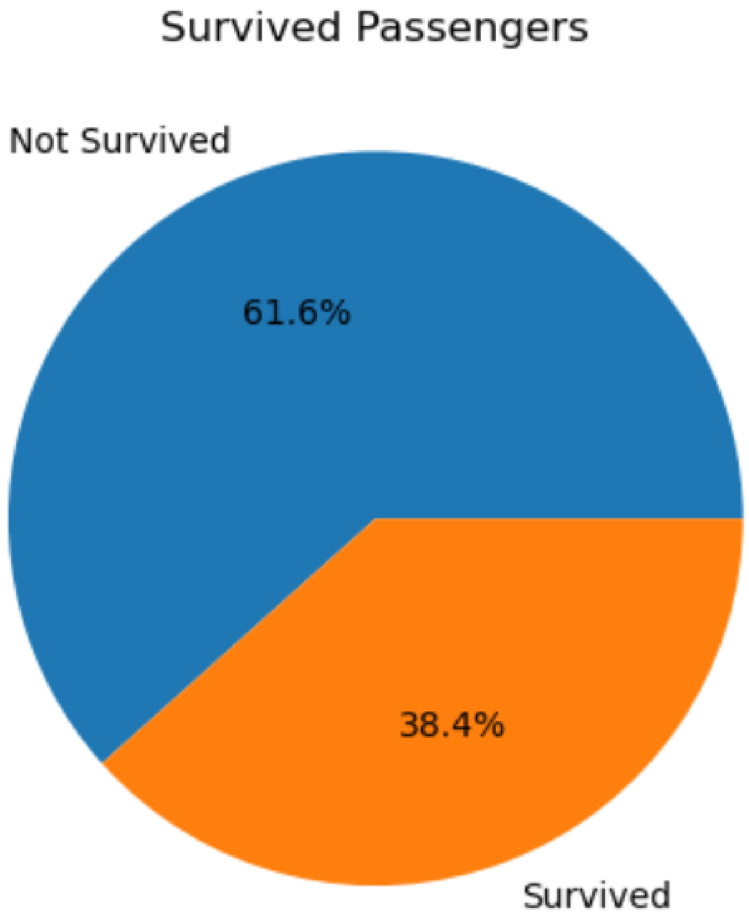
Week 14 Assignment (1)

- **Problem 1:** Draw the distribution according to 'Age' as a histogram.
  - Bins can be set freely.



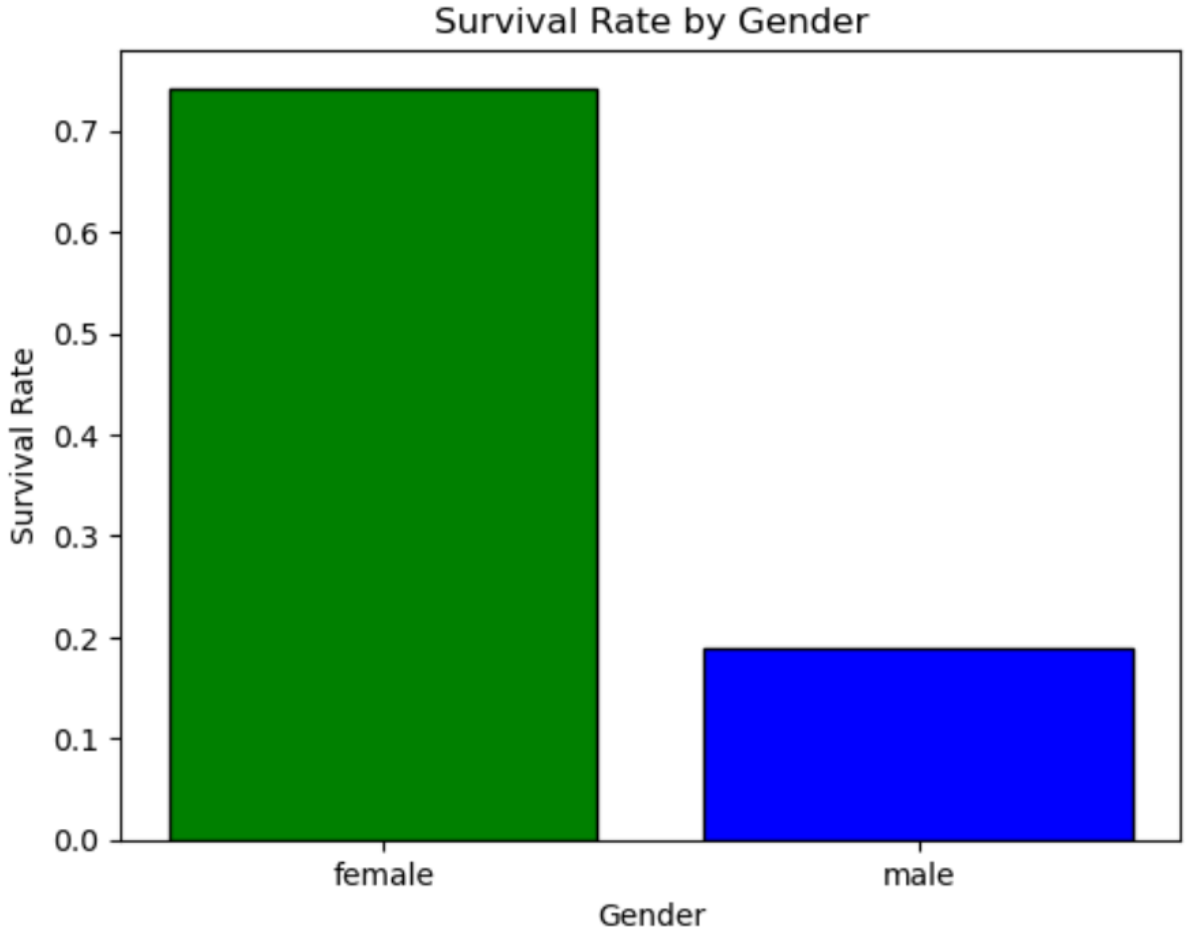
Week 14 Assignment (1)

- **Problem 2:** Visualize the ratio of survivors("Survived==1") and non-survivors ("Survived==0") using the pie chart



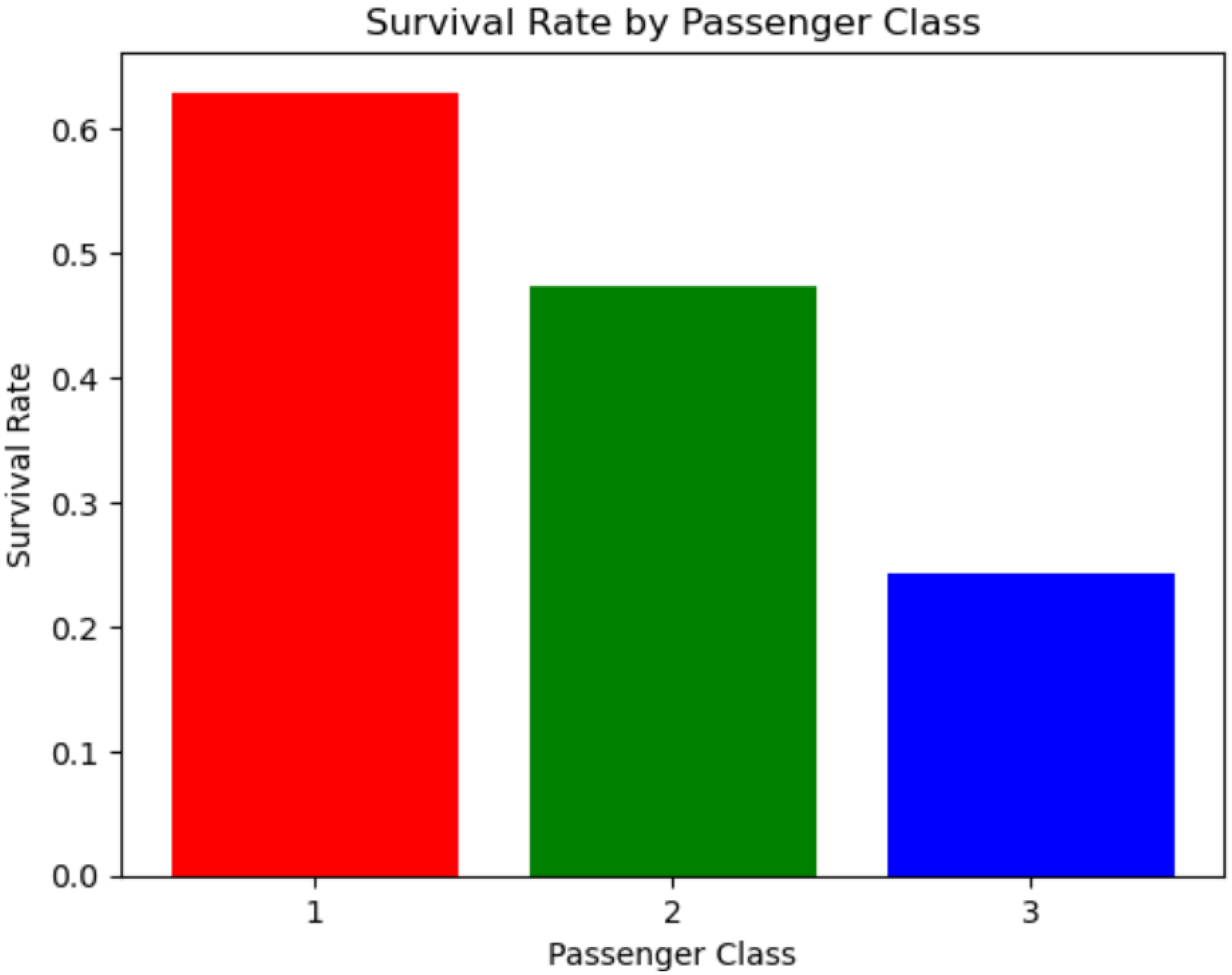
Week 14 Assignment (1)

- **Problem 3:** Visualize the survival rate by gender ('Sex') using the bar chart
  - hint) using the groupby()



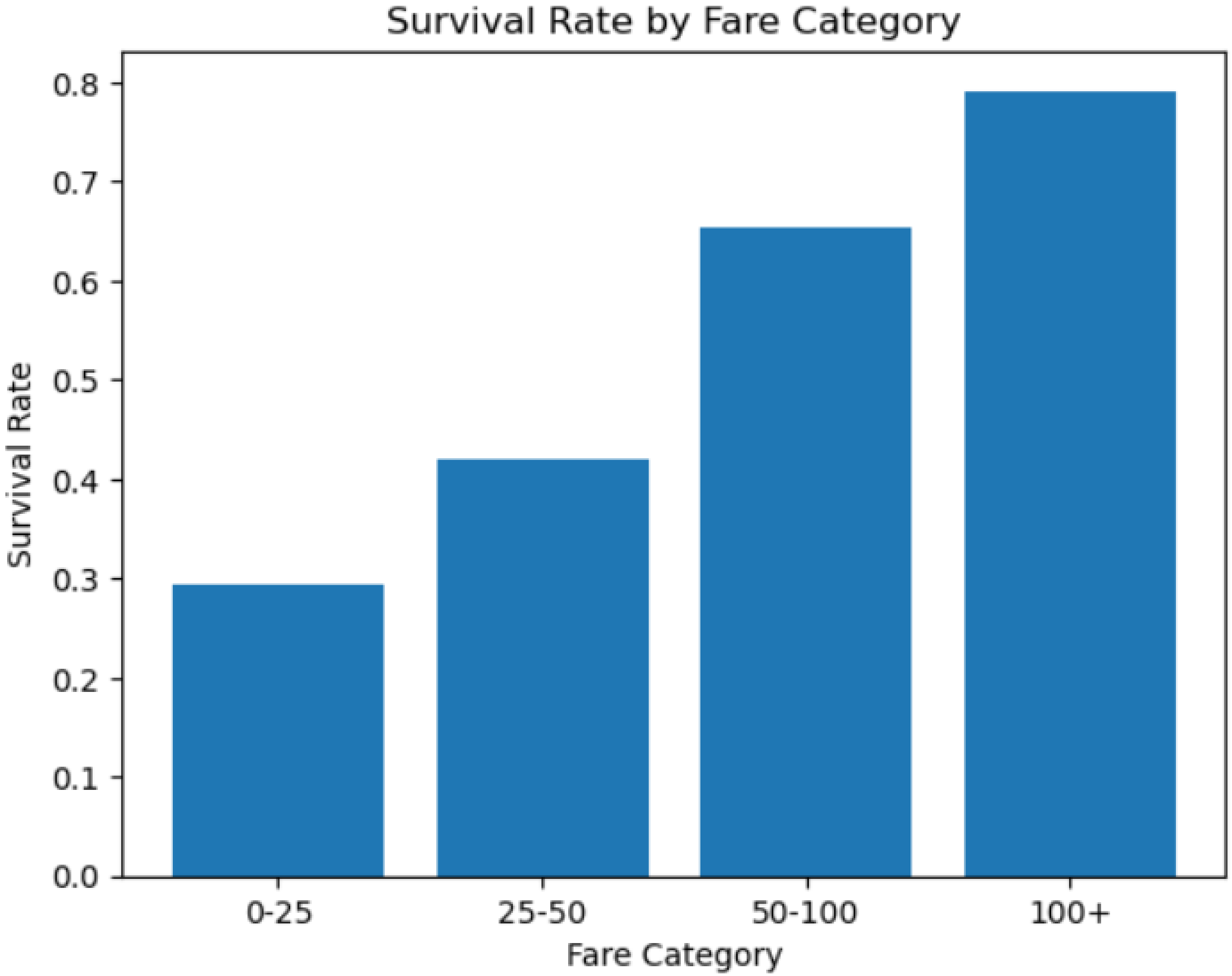
Week 14 Assignment (1)

- **Problem 4:** Visualize the survival rate by passenger class (‘Pclass’) using the bar chart
  - hint) using the groupby()



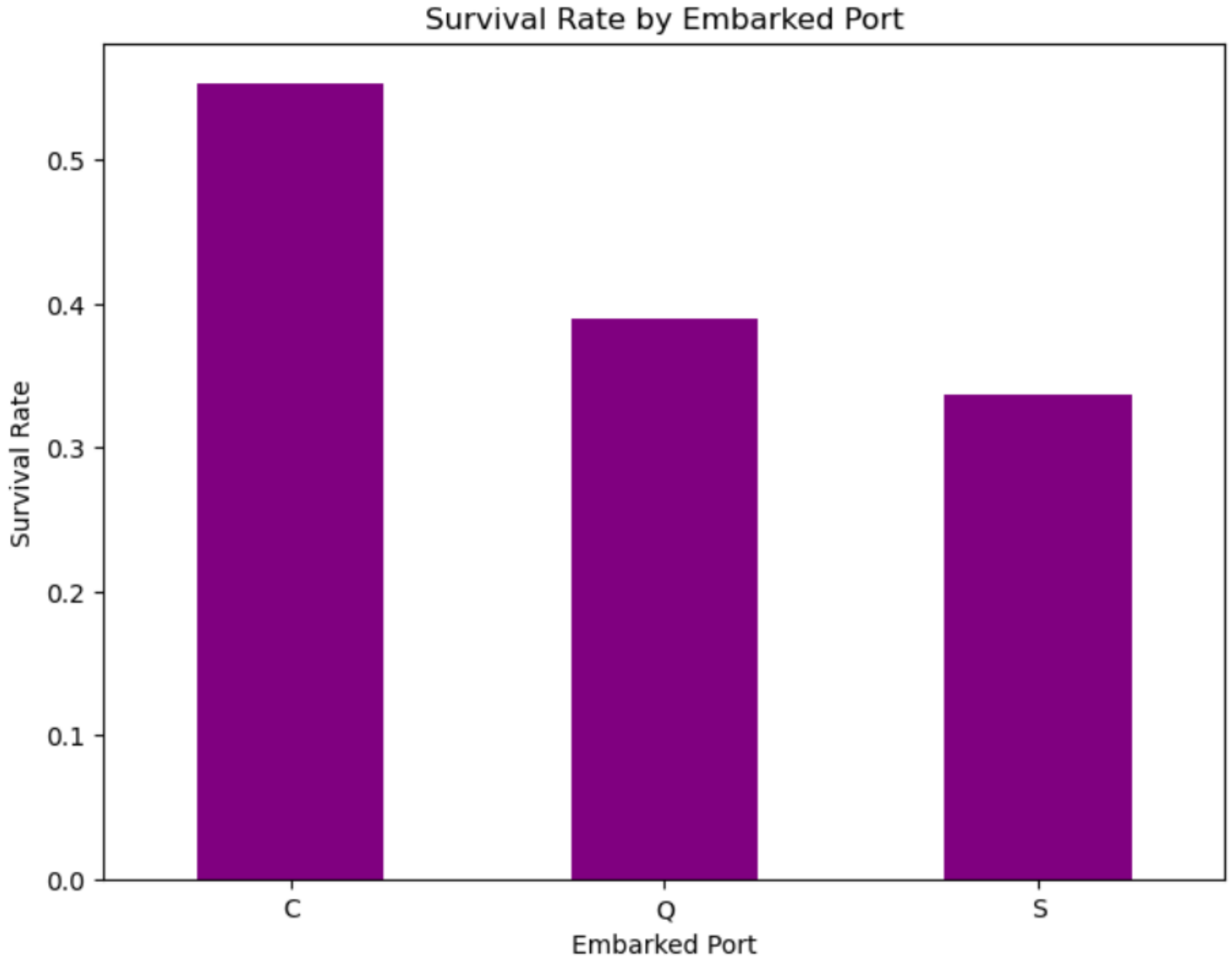
Week 14 Assignment (1)

- **Problem 5:** Visualize the survival rate by fare ***category*** using the bar chart
  - Divide ‘Fare’ into four categories (bins)
  - hint) using the cut() and groupby()



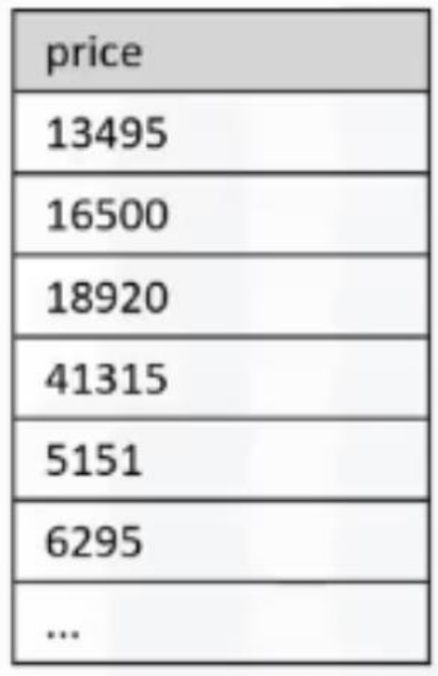
Week 14 Assignment (1)

- **Problem 6:** Visualize the survival rate by embarked port ('Embarked') using the bar chart
  - hint) using the groupby()

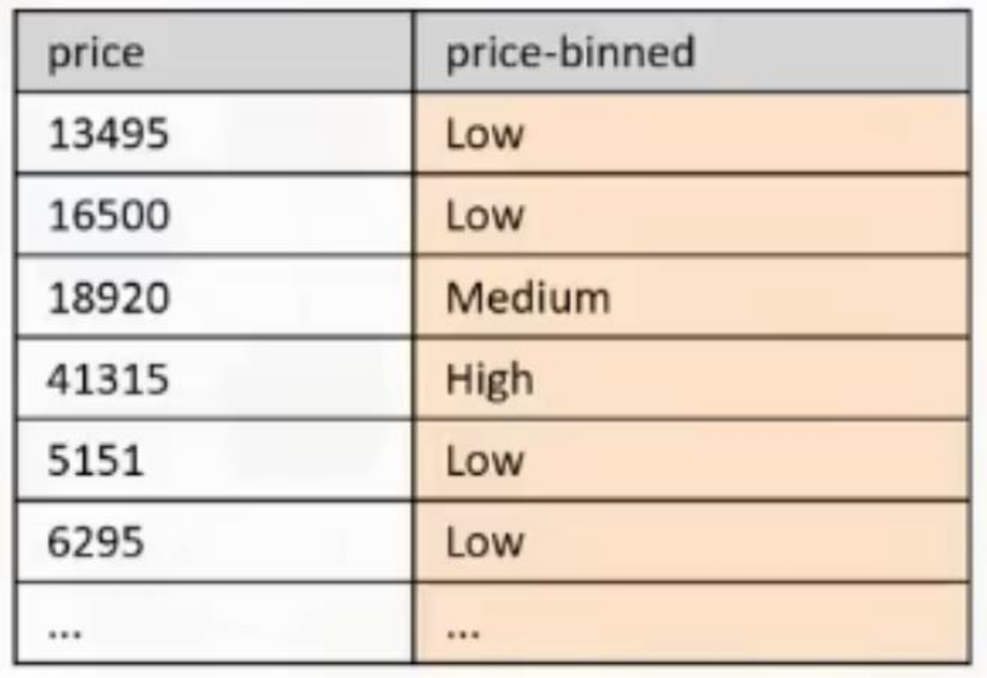


Data Binning

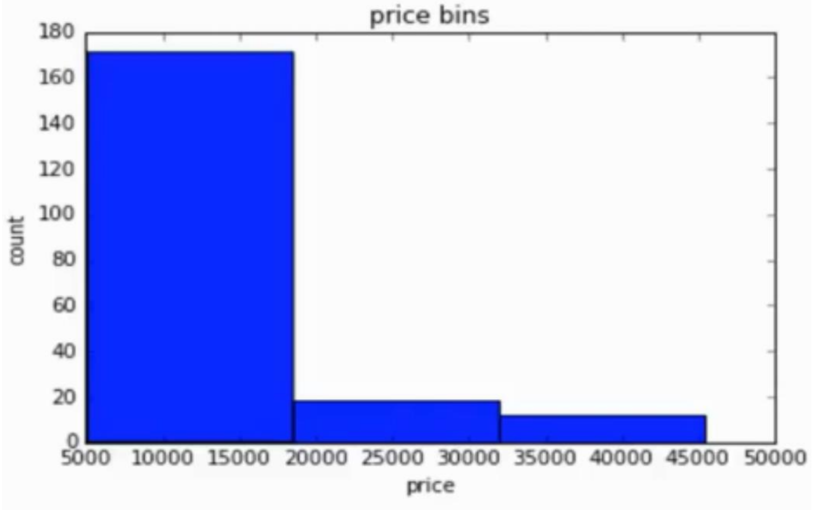
- **Binning**
  - Grouping of values into “bins”
  - Converts numeric into categorical variables



price
13495
16500
18920
41315
5151
6295
...

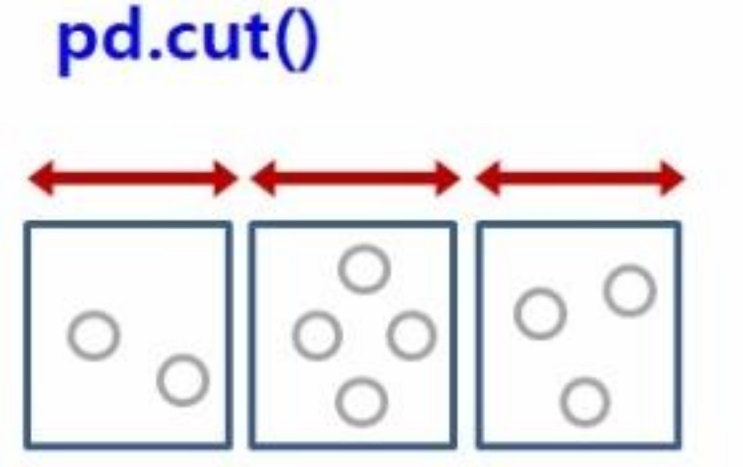


price	price-binned
13495	Low
16500	Low
18920	Medium
41315	High
5151	Low
6295	Low
...	...



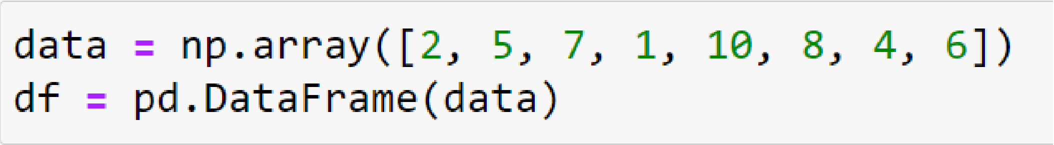
Data Binning

- cut() : binning data into user-defined bins (length buckets).

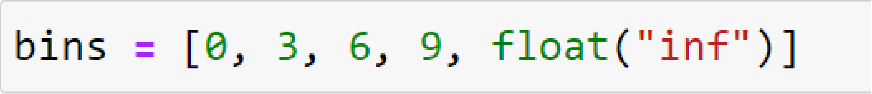


Data Binning

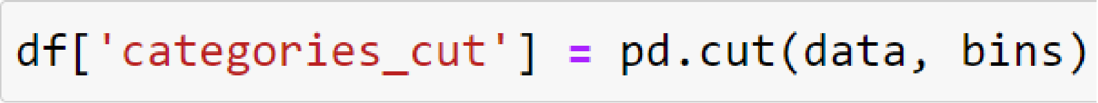
- cut()



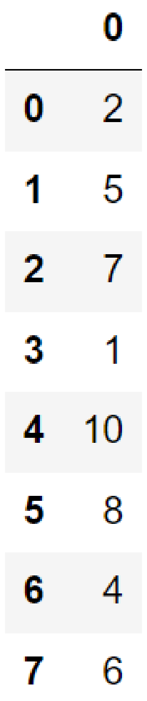
```
data = np.array([2, 5, 7, 1, 10, 8, 4, 6])
df = pd.DataFrame(data)
```



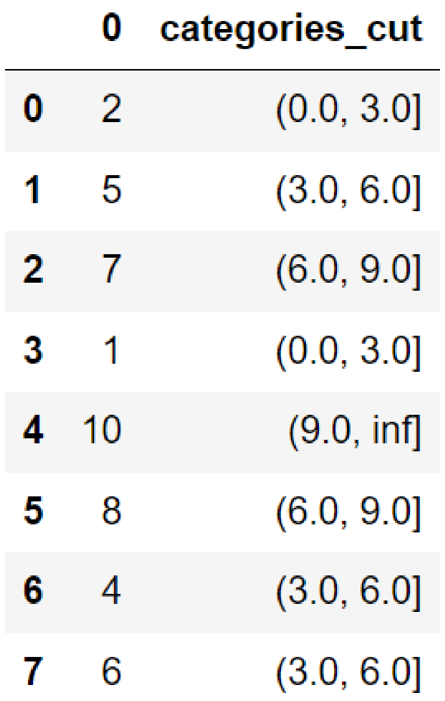
```
bins = [0, 3, 6, 9, float("inf")]
```



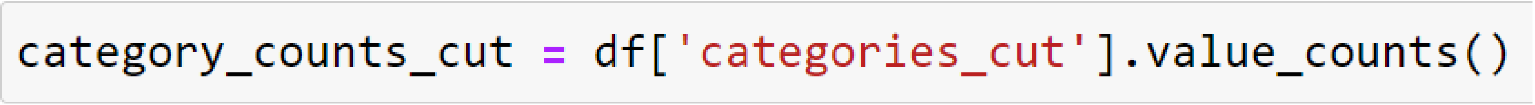
```
df['categories_cut'] = pd.cut(data, bins)
```



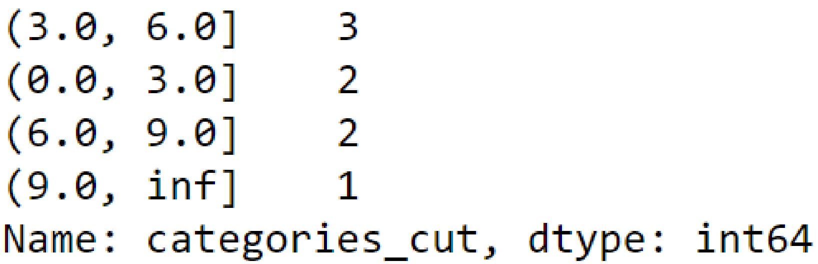
	0
0	2
1	5
2	7
3	1
4	10
5	8
6	4
7	6



	0	categories_cut
0	2	(0.0, 3.0]
1	5	(3.0, 6.0]
2	7	(6.0, 9.0]
3	1	(0.0, 3.0]
4	10	(9.0, inf]
5	8	(6.0, 9.0]
6	4	(3.0, 6.0]
7	6	(3.0, 6.0]



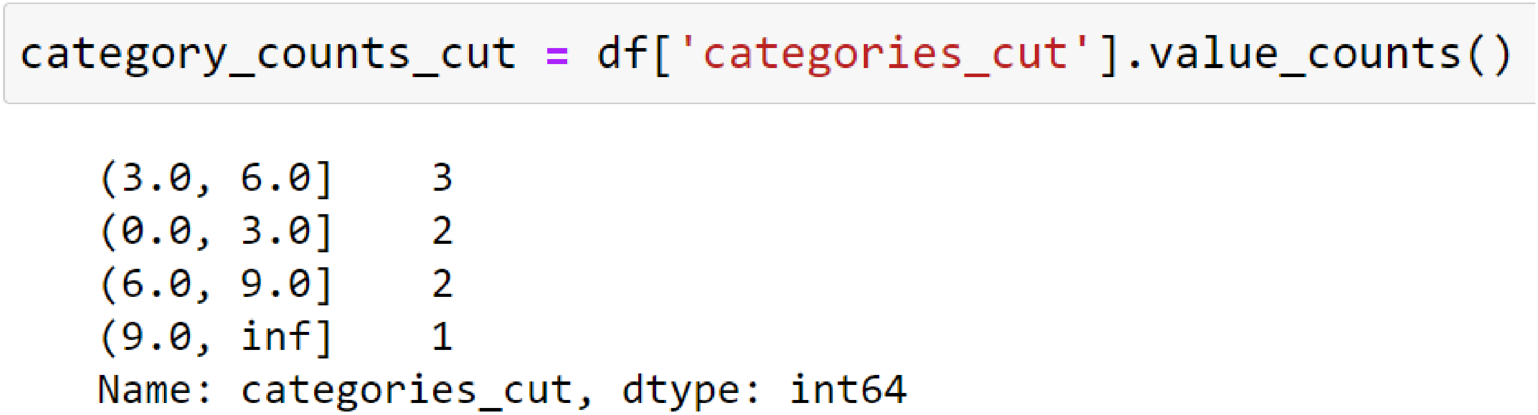
```
category_counts_cut = df['categories_cut'].value_counts()
```



```
(3.0, 6.0]    3
(0.0, 3.0]    2
(6.0, 9.0]    2
(9.0, inf]    1
Name: categories_cut, dtype: int64
```

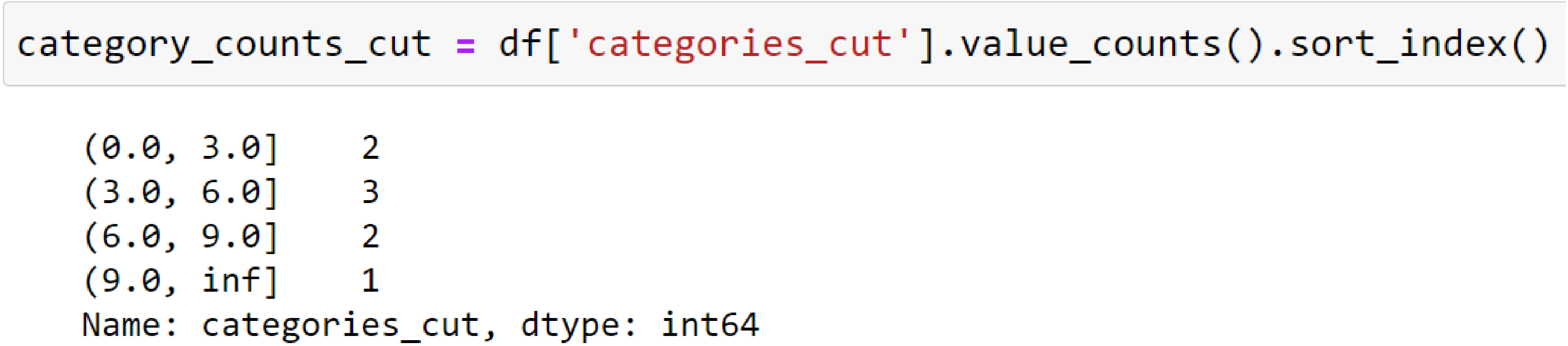
Data Binning

- cut()



```
category_counts_cut = df['categories_cut'].value_counts()
```

```
(3.0, 6.0]    3
(0.0, 3.0]    2
(6.0, 9.0]    2
(9.0, inf]    1
Name: categories_cut, dtype: int64
```



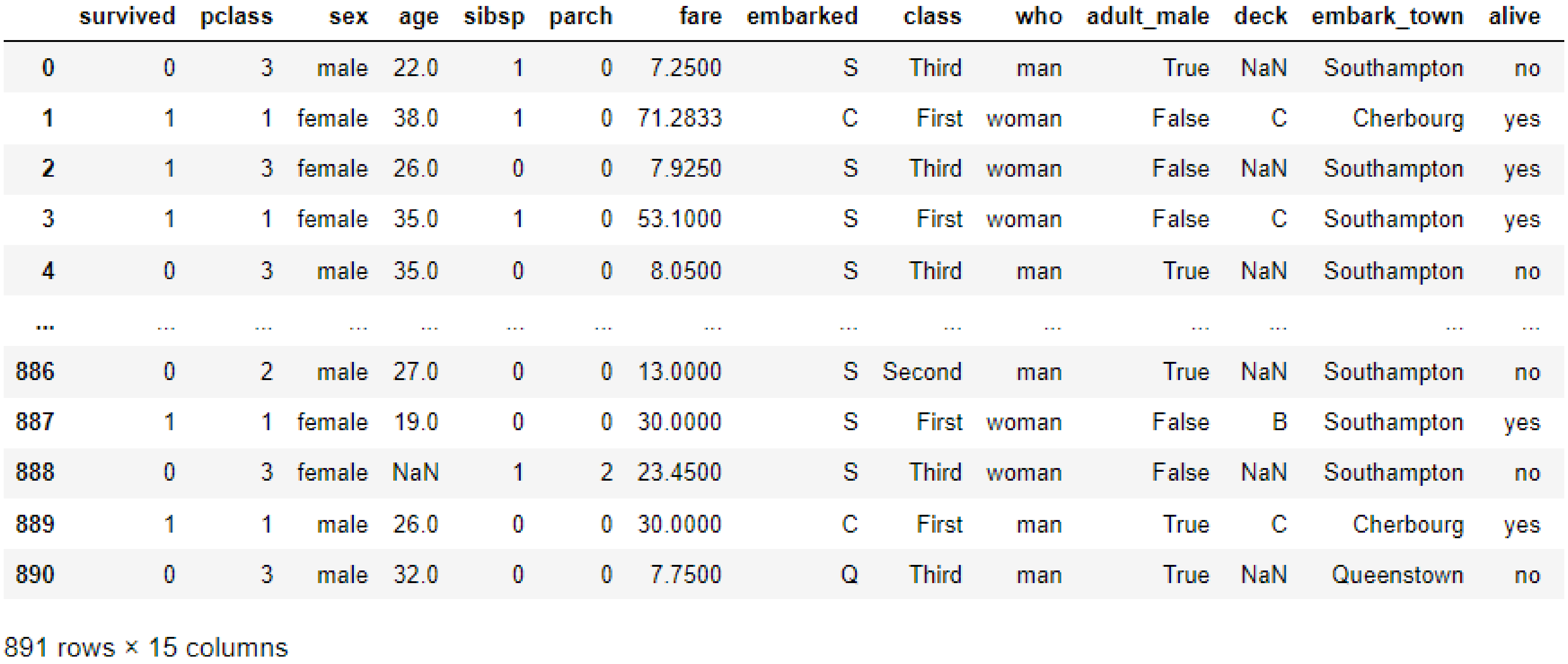
```
category_counts_cut = df['categories_cut'].value_counts().sort_index()
```

```
(0.0, 3.0]    2
(3.0, 6.0]    3
(6.0, 9.0]    2
(9.0, inf]    1
Name: categories_cut, dtype: int64
```

Week 14 Assignment (2)

Week 14 Assignment (2)

- **Problem 1: Loading the 'titanic' dataset from the online repository provided by the seaborn library**
  - Requires internet

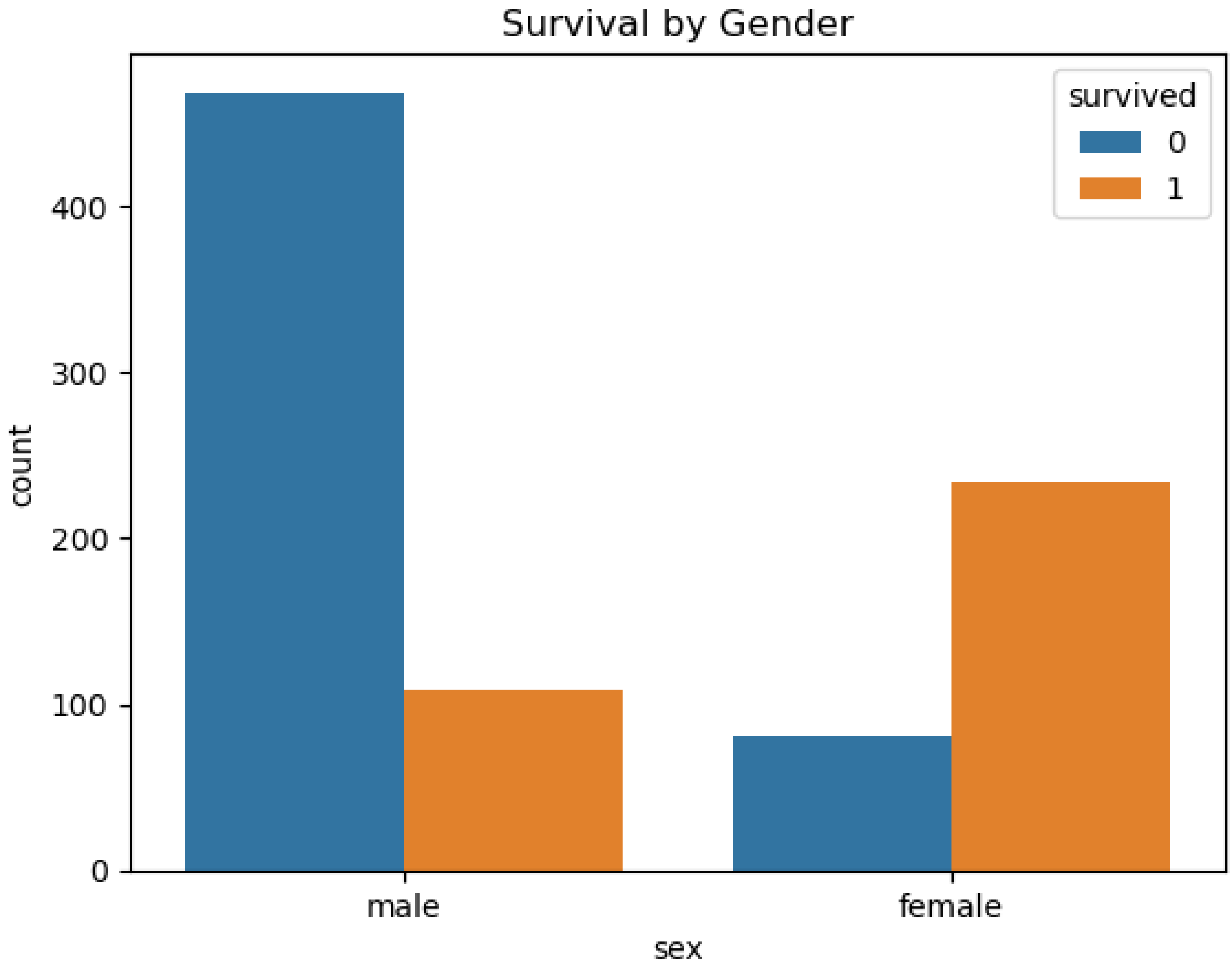


	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no

891 rows × 15 columns

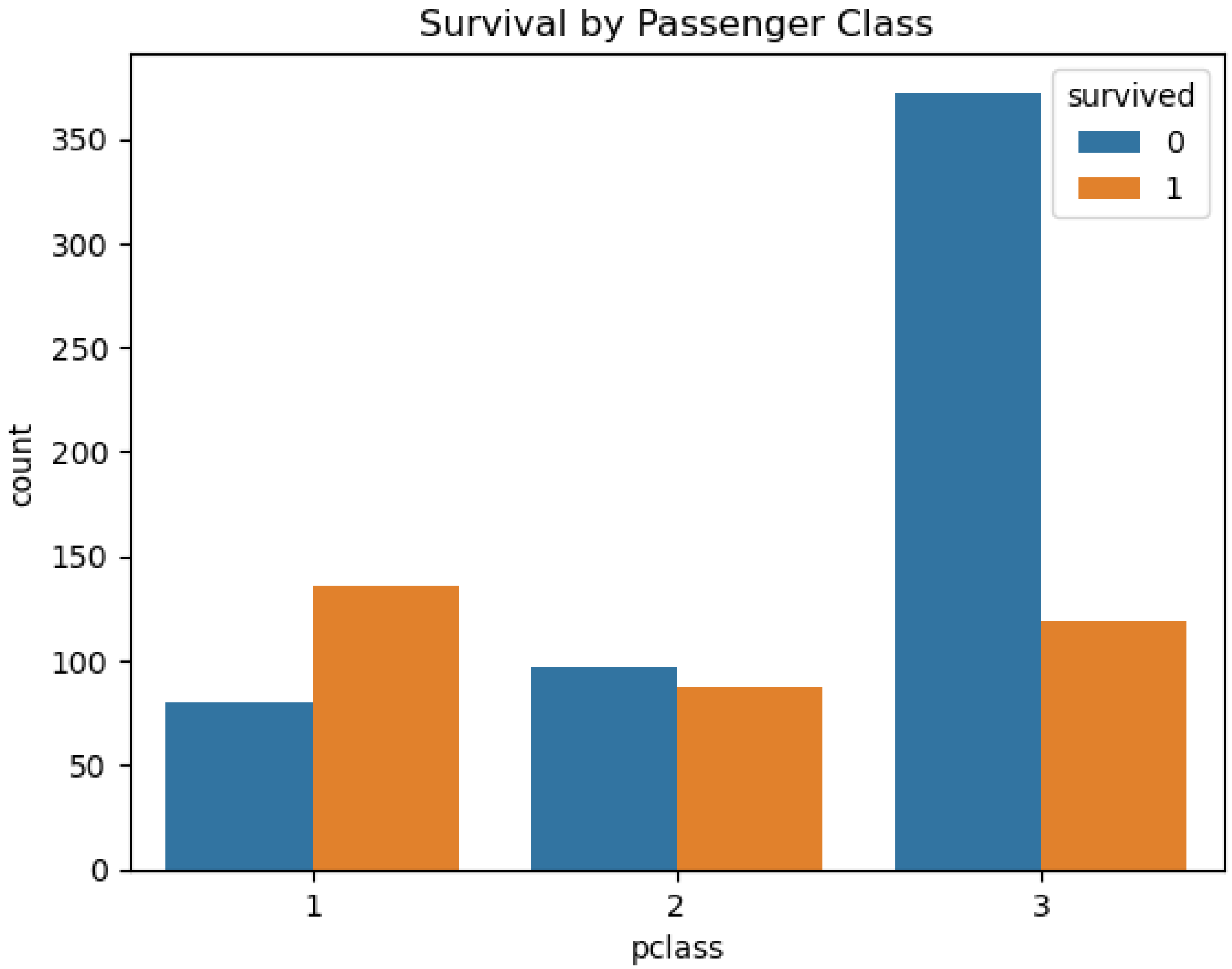
Week 14 Assignment (2)

- Problem 1: Visualize the number of survivor by gender



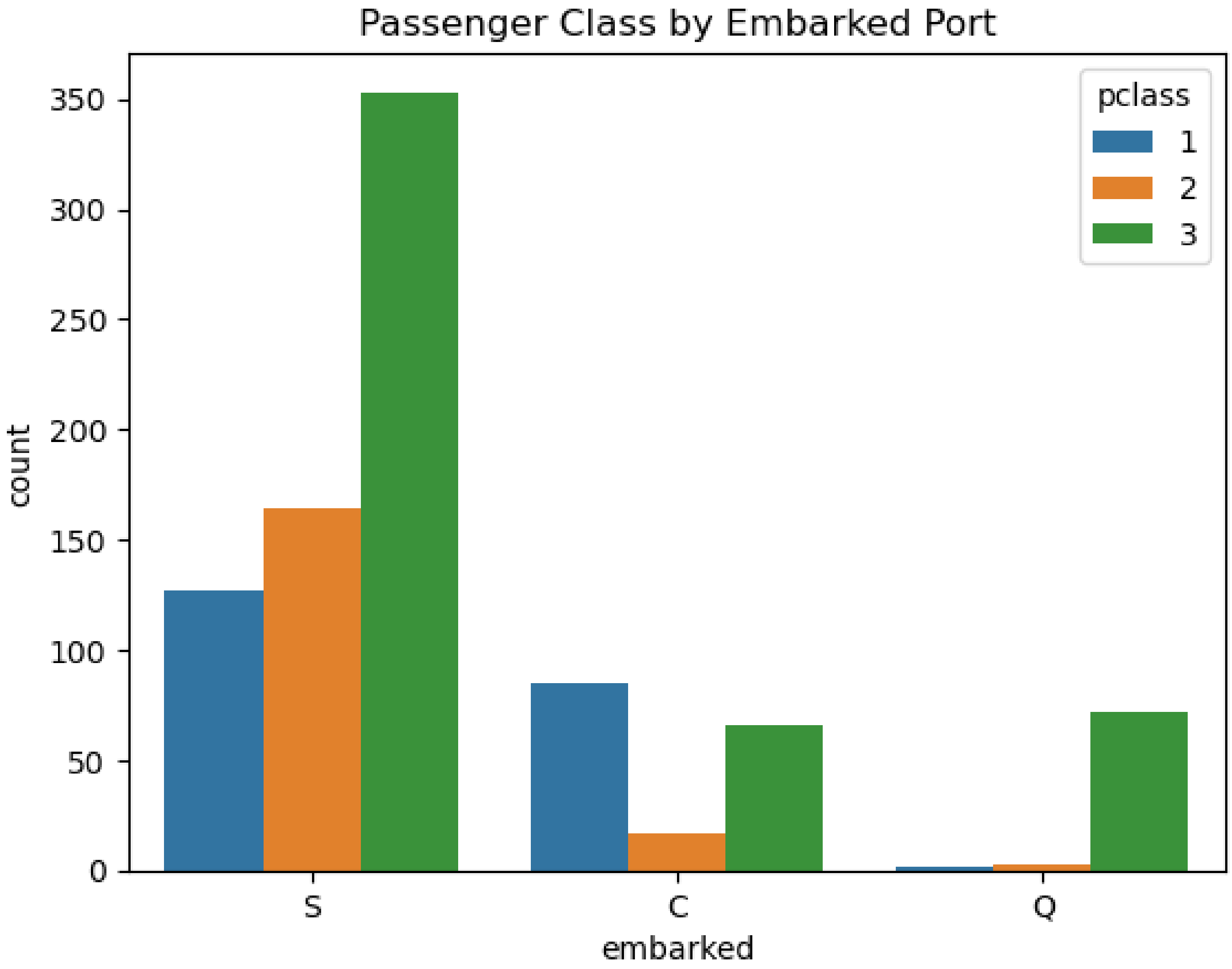
Week 14 Assignment (2)

- Problem 2: Visualize the number of survivor by passenger class



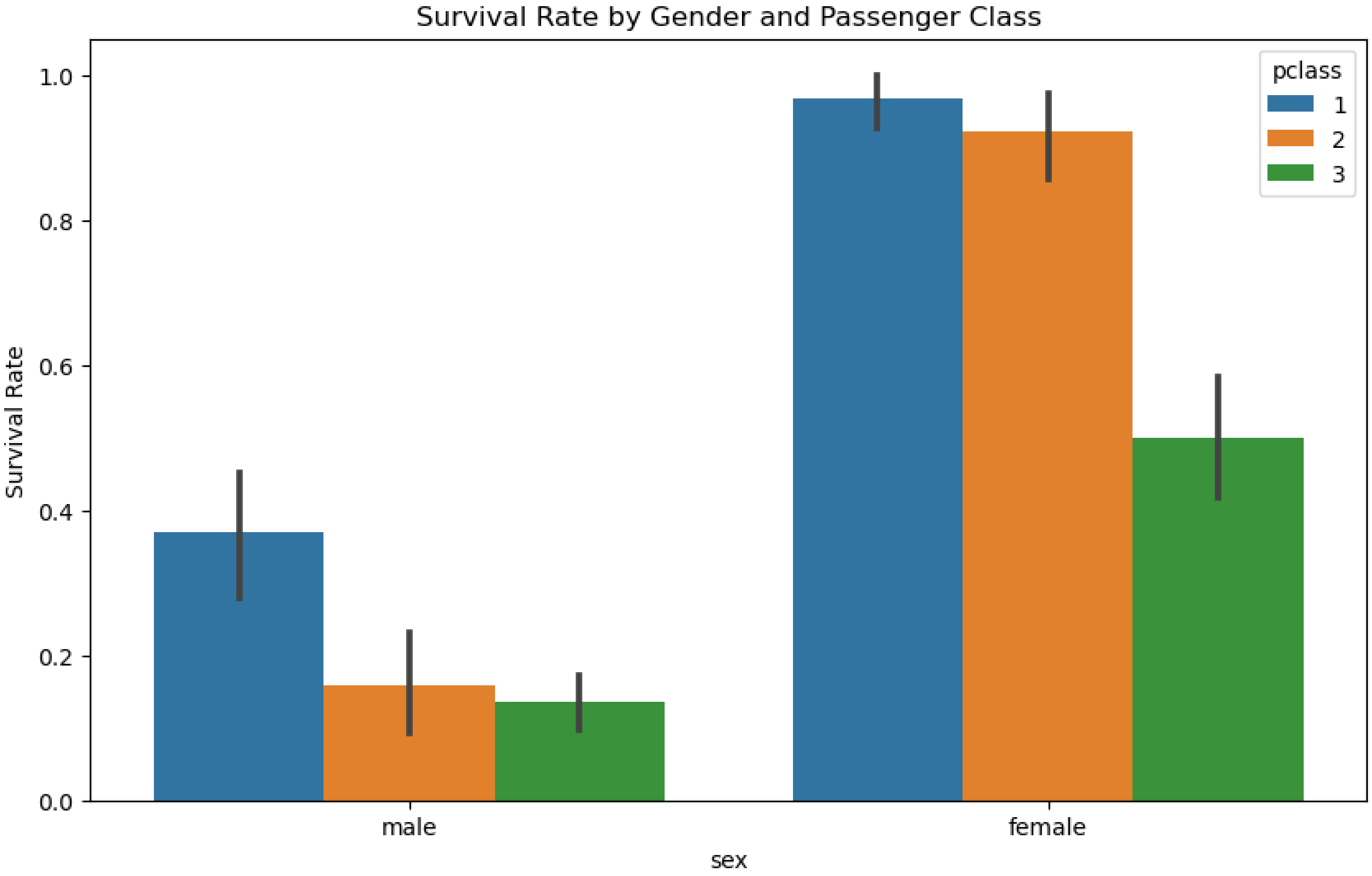
Week 14 Assignment (2)

- **Problem 3: Visualize the number of people per passenger class by embarked port**



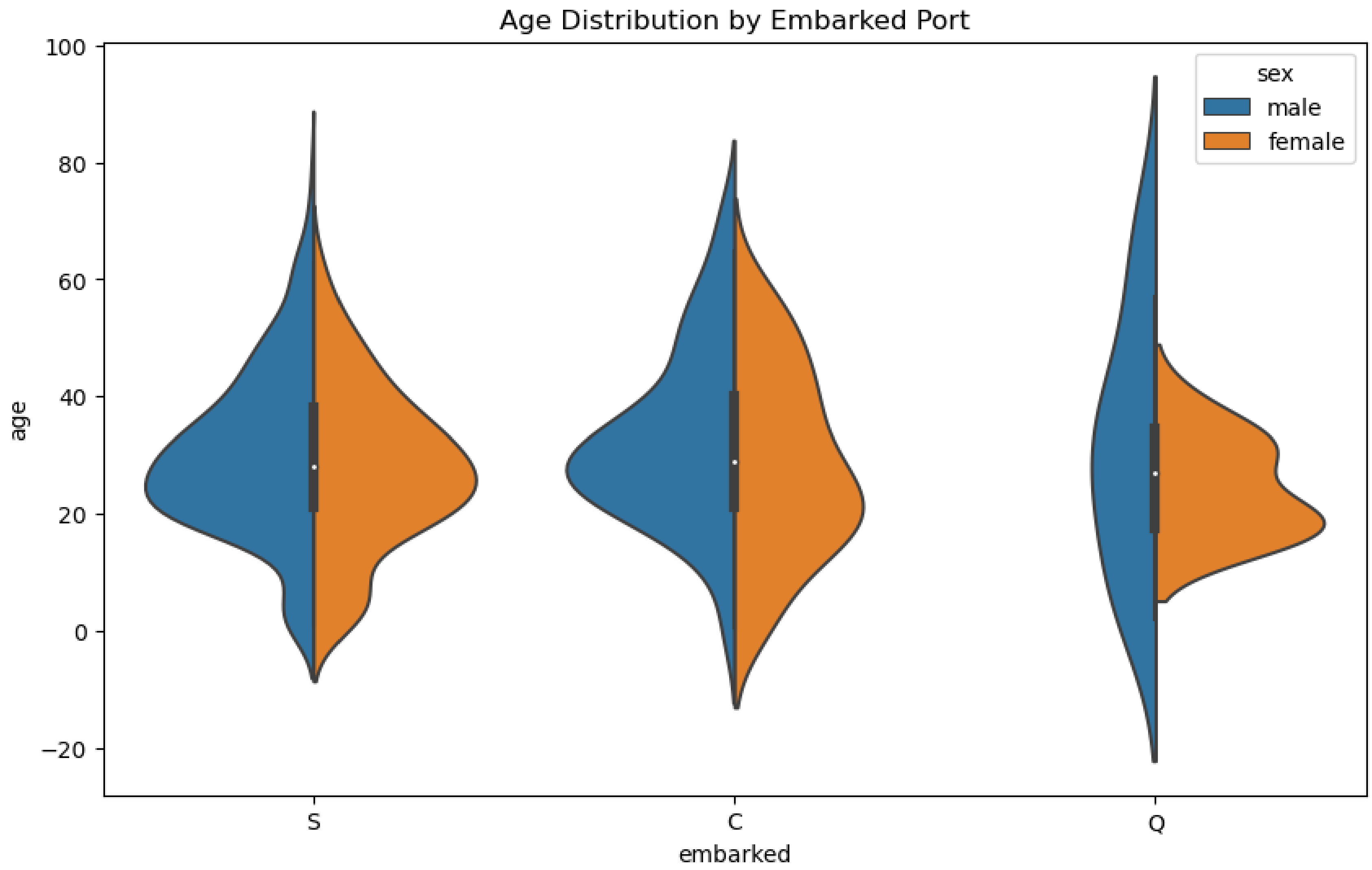
Week 14 Assignment (2)

- Problem 4: Visualize survival rate by gender and passenger class



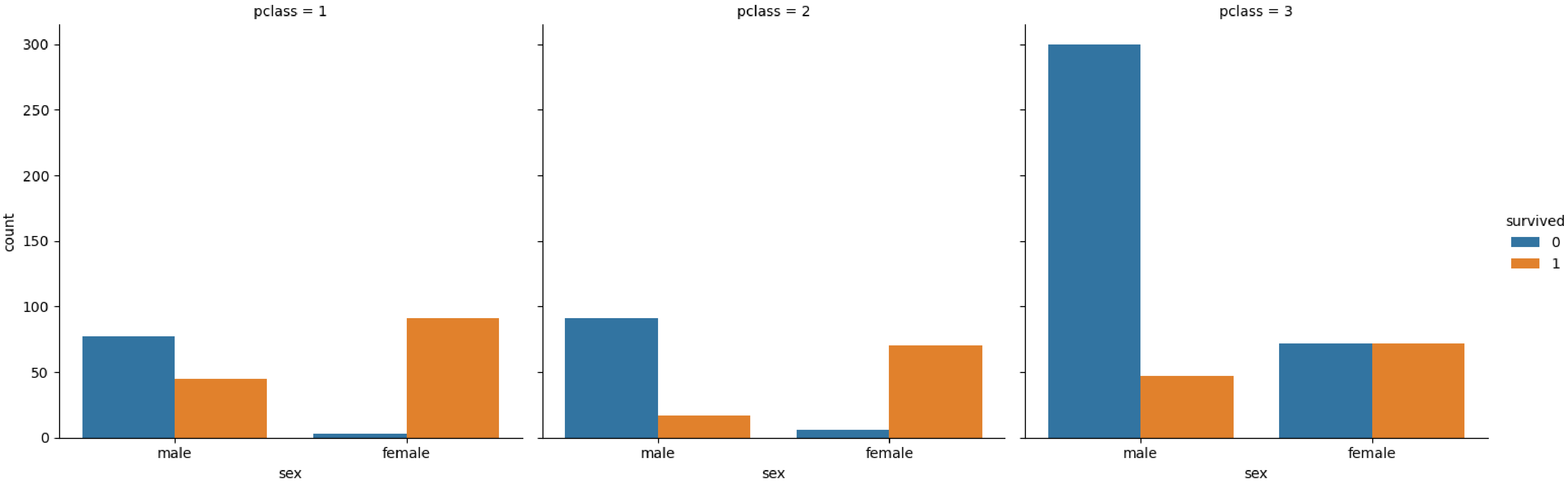
Week 14 Assignment (2)

- **Problem 5: Visualize age distribution by embarked port and gender.**



Week 14 Assignment (2)

- **Problem 6: Visualize the survival by gender and passenger class**


pclass = 1
pclass = 2
pclass = 3
count
male
female
sex
survived